



Research article

Hybrid deep representation learning-based adversarial attack detection framework for secure communication in 6G-assisted IoT systems

Mohammed A. AlAqil¹, Amal K. Alkhalifa², Sultan Alahmari³, Nadhem NEMRI⁴, Saied Alshahrani⁵, Sulaiman Alamro⁶, Sultan Almutairi^{7,*} and Achraf Ben Miled⁸

¹ Department of Electrical Engineering, College of Engineering, King Faisal University, AlAhsa, Saudi Arabia

² Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia

³ King Abdul Aziz City for Science and Technology (KACST), Cybersecurity Institute, Riyadh, Saudi Arabia

⁴ Department of Information Systems, Applied College at Mahayil, King Khalid University, Saudi Arabia

⁵ Department of Computer Science and Artificial Intelligence, College of Computing and Information Technology, University of Bisha, Bisha 61922, Saudi Arabia.

⁶ Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia

⁷ Department of Computer Science, Applied College, Shaqra University, Shaqra 15526, Saudi Arabia

⁸ Department of Computer Science, College of Science, Northern Border University, Arar 73213, Saudi Arabia

* **Correspondence:** Sultan Almutairi; sultan@su.edu.sa.

Abstract: Internet of Things (IoT) networks play a significant role in sixth-generation (6G) networks due to higher throughput, lower latency, ultra-reliability, and dense connectivity. Artificial intelligence (AI), particularly deep learning (DL) and machine learning (ML), offers an effective means to advance and implement next-generation radio communication technologies. However, these approaches are vulnerable to adversarial attacks, which can significantly degrade system performance and lead to inaccurate predictions. This outcome is highly undesirable for a reliable and ubiquitous IoT-based communication system. This paper presents a hybrid neural network framework for adversarial machine learning attack detection using ridge regression (HNNF-AMLADRR) in 6G network-assisted

intelligent IoT systems. The main objective is to develop an effective detection framework for adversarial machine learning (AML) attacks, thereby improving the security of intelligent systems in 6G and IoT environments against malicious perturbations. First, the min-max scaler normalization method is employed in data preprocessing to transform the input data into a suitable format. Moreover, ridge regression is primarily used for feature reduction to remove irrelevant or redundant features. For adversarial attack detection, the proposed HNNF-AMLADRR model designs a hybrid model of a self-attention-based restricted Boltzmann machine with a recurrent convolutional neural network technique. Finally, the model parameters are fine-tuned using the Adafactor optimizer to enhance classification performance. The experimental evaluation of the HNNF-AMLADRR algorithm was done using the AML dataset. Empirical outcomes indicated an enhanced performance of the HNNF-AMLADRR model compared to recent approaches.

Keywords: adversarial attacks; machine learning; ridge regression; sixth-generation network; Internet of Things; self-attention

Mathematics Subject Classification: 68T07, 68T45

1. Introduction

Sixth-generation (6G) networks are regarded as the next development in wireless technology, succeeding the current fifth-generation (5G) communication systems. The predicted advantages of 6G over preceding generation networks include energy efficiency, data capacity, latency, speed, and reliability. In particular, the peak data rate in 6G communication is predicted to be 1 terabit per second (Tbps), nearly 10 times faster than today. 6G systems are projected to provide significantly larger data capacity, thus permitting data-intensive applications and more interconnected devices. This detail responds to the constant upsurge of the Internet of Things (IoT) in the upcoming years. Regarding latency, the 6G systems are predicted to surpass 5G networks by reducing the transmission delay to less than one millisecond (MS). Unprecedentedly lower latency in 6G communication can enable innovative applications with real-world responses and higher accuracy, namely in remote surgery, smart city, and autonomous vehicle applications.

Adversarial machine learning (AML) is a current research domain that examines potential security concerns associated with the employment of machine learning (ML) models in existing artificial intelligence (AI)-driven systems, alongside defensive strategies to safeguard ML systems from those attacks [1]. Adversarial attacks are a key issue for ML, comprising methods intended to trick ML models by subtly modifying input data. These attacks exploit vulnerabilities to cause misclassifications or other unintentional behaviors in the model [2]. Essentially, intruders make specific input alterations to influence the model's decision-making processes, making it an increasing threat in the AI field. With AI and ML methods being widely utilized in various domains, AML attacks have garnered growing interest from security experts and developers [3]. Critical industrial systems, such as healthcare systems and autonomous transportation, employ ML techniques. If compromised, these systems can lead to catastrophic health and safety incidents [4,5]. For instance, if an autonomous self-driving car utilizing a traffic sign detection system is compromised by adversaries, it could falsely identify stop signs as speed limit signs. This might result in road accidents. Likewise, if a medical system is compromised by rival healthcare service providers, fake insurance companies, or attackers, they could misdiagnose an illness

and provide inappropriate treatments to patients [6]. Novel progress in computer vision (CV) has trained classifiers closer to the forefront of critical security systems as well.

AML attacks are a set of techniques employed to weaken the precision, reliability, or security of ML systems [7]. AML attacks are launched by cybercriminals with diverse motives, including information theft, affecting privacy and confidentiality, or influencing decision-making processes of ML systems. AML attacks are launched against a wide array of ML methods [8]. The success of AML attacks relies on numerous aspects, like the quality and type of targeted ML models, the complexity of the attack method, and the malicious actors' knowledge and resources. Based on the attack taxonomy, they are categorized into evasion, poisoning, extraction, and inference attacks [9]. Evasion attacks in AML involve adversaries crafting modified inputs to induce misclassification by a machine learning model, without altering the fundamental properties or inherent nature of the data [10].

Data poisoning attacks in AML occur when malicious actors intentionally alter the data used to train an ML model [11]. This can lead to the model learning wrong patterns, making it unfair or prone to errors. Model extraction threats in AML are a type of threat in which attackers focus on illegally extracting information from an ML system [12]. AML inference attacks refer to situations where intruders attempt to collect private information regarding the input data employed by the ML system. The domain of AML also includes the investigation of defenses against these attacks [13]. Scholars are developing several methods to identify and reduce the influence of adversarial examples, including techniques to detect dubious inputs, train models to be more robust against manipulations, and create more secure AI models. The continuing effort to understand and counteract these attacks is vital for ensuring the trustworthiness and reliability of AI in several key applications [14]. Recently, deep reinforcement learning (DRL) has gained attention in adversarial ML for emerging adaptive attacks and defense plans. DRL methods, like deep deterministic policy gradient (DDPG), proximal policy optimization (PPO), and deep Q-network (DQN), have been applied to improve the reliability of AI-based systems. However, existing DRL models often suffer from high computational complexity and limited generalization despite hidden adversarial attacks. Figure 1 represents the general architecture of adversarial attack detection for 6G-assisted IoT.

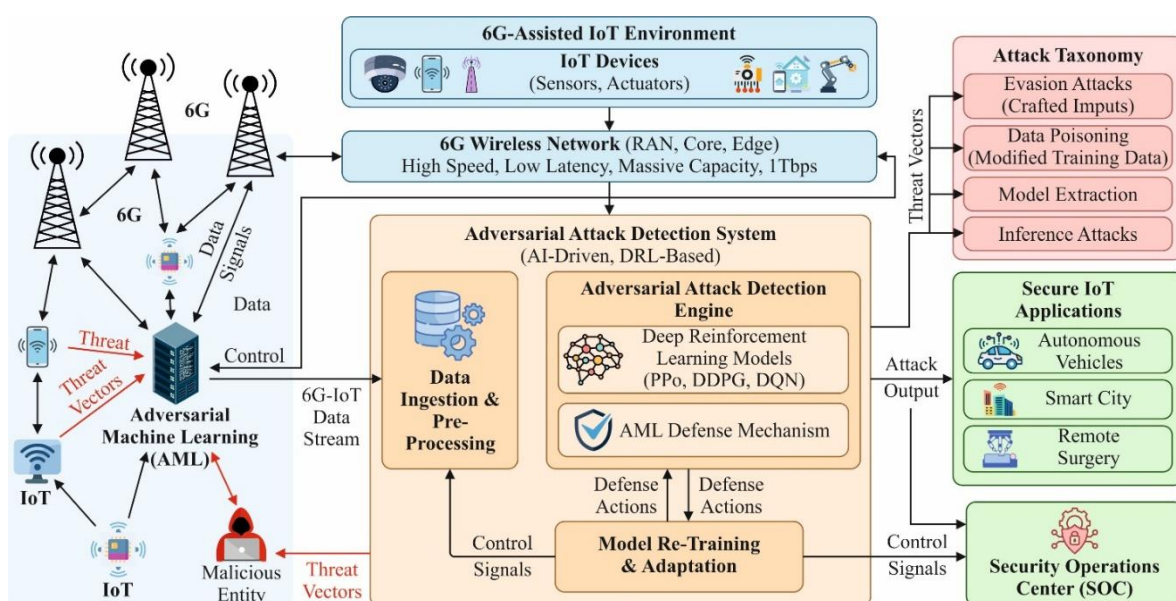


Figure 1. General architecture of adversarial attack detection for 6G-assisted IoT.

This paper presents a hybrid neural network framework for adversarial machine learning attack detection using ridge regression (HNNF-AMLADRR) in 6G network-assisted intelligent IoT systems. The proposed framework consists of a preprocessing stage, a feature subset selection process, attack classification using a hybrid neural network, and a parameter fine-tuning mechanism. Experimental analysis confirms the efficacy of the proposed HNNF-AMLADRR architecture. The core contributions of this article are summarized as follows:

- Development of an effective ridge regression–based feature subset selection method to recognize and eradicate irrelevant, noisy, and redundant features from the dataset, thereby enhancing the input data quality and improving the overall learning proficiency of the proposed framework.
- Design of a hybrid neural network (NN) approach of self-attention-based restricted Boltzmann machine (RBM) with recurrent convolutional neural network (RCNN) technique to efficiently learn spatial-temporal patterns and accurately perform adversarial attack detection and classification in intelligent IoT environments.
- Adoption of the Adafactor optimization algorithm to fine-tune the model parameters, which helps to improve convergence efficiency, reduce computational overhead, and significantly enhance the classification performance of the hybrid NN method.
- The experimental evaluation of the HNNF-AMLADRR algorithm occurs using the adversarial machine learning dataset across three different applications, namely DNS tunneling, platooning, and remaining useful life (RUL). The comparative experimental study demonstrates that the proposed model achieves superior detection accuracy compared to several recent algorithms.

2. Literature review

This section offers a detailed review of the existing literature on adversarial attack detection methods and their applications in intelligent systems. In [15], AML is related to the domain of research that entails testing and enhancing ML systems by presenting adversarial attack methods. In cybersecurity, AML has a major ability to aid in identifying and defending against attacks such as malware, cyber threats, and fraud detection. Nevertheless, AML also encounters several difficulties, such as low efficacy in creating adversarial samples. There is an active relationship between adversarial attacks and defenses, with intruders constantly creating novel methods, and defenders continuously requiring enhancement of their defense tactics. Bommana et al. [16] introduced a new AI-driven DL approach to identify adversarial attacks in IoT infrastructures, enhancing data preprocessing, feature extraction, and classification using holistic methods. A three-step filtering method featuring adaptive weights has been employed to improve the effectiveness of data preprocessing. Barik et al. [17] presented a weighted conditional stepwise adversarial network alongside a PSO (WCSAN-PSO) model for identifying adversarial attacks in intrusion detection system (IDS). Moreover, PSO enhanced the discriminator and generator parameters in WCSAN to advance IDS's adversarial training. Yuan et al. [18] examined a novel IDS framework that maximizes the functionalities of IDS against adversarial attacks by merging traditional ML and DL approaches. Then, they utilized the low attack transferability among ML and DL techniques to identify a strong ML methodology that helps to determine AE's maliciousness. Petihakis et al. [19] discussed emerging threats related to adversarial AI and outlined the growth of AIAS. AIAS is an inclusive, AI-based

security platform created to improve the resilience of AI systems against such risks. Moreover, AIAS features advanced mechanisms for simulating, detecting, mitigating, and deceiving threats through adversarial defense methods, sophisticated honeypots, and attack detection modules. Sheikh et al. [20] presented ML-driven smart attack recognition approaches, which have progressed due to setbacks in conventional signature-based methods to identify zero-day attacks and complex nature attacks.

Alabsi et al. [21] suggested an IDS that relies on a conditional tabular GAN (CTGAN) for identifying DoS and DDoS threats in IoT systems. The CGAN-enabled IDS employs a generator network to generate artificial traffic that imitates a genuine traffic pattern, while discriminator networks learn to distinguish between genuine and hostile traffic. Dhamija and Bansal [22] developed a novel defensive approach to enhance the strength of detection and classification techniques against many adversarial threats. This research presented a dual-step defensive method through a distinctive adoption of removing strong feature representations, model-agnostic adversarial feature learning, and attention feature fusion. Table 1 represents the comparative analysis of existing adversarial attack detection on single DL methods or conventional hybrid algorithms, with limited attention to incorporating self-attention mechanisms, hybrid RCNN-RBM methods, and optimization approaches. These limitations motivate the proposed HNNF-AMLADRR method.

Table 1. Overview of related works on adversarial attack detection in cybersecurity and IoT environments.

Authors	Objectives	Findings	Methods	Research gap
Xu et al. [15]	To analyze adversarial ML approaches in cybersecurity and study distinct varieties of threats and defense mechanisms.	Growing vulnerability of AI systems.	Survey-based analysis of adversarial attacks.	Absence of an implementation-based framework for real-time adversarial attack recognition in IoT or 6G networks.
Bommana et al. [16]	To address adversarial attacks in IoT environments utilizing DL-based AI approaches.	Accuracy of 0.92 and 0.95.	CNN and deep learning models.	Does not consider hybrid neural models or feature selection (FS) mechanisms to enhance recognition efficacy.
Barik et al. [17]	To improve an adversarial attack recognition framework utilizing an optimized weighted conditional stepwise adversarial model.	Accuracy of 99.36%.	PSO, least absolute shrinkage and selection operator (LASSO), principal component analysis (PCA), and support vector classification (SVC) models.	The framework suffers from high computational complexity and restricted adaptability to large-scale IoT systems.
Yuan et al. [18]	To develop the adversarial robustness of DL-based IDS.	Achieved better results.	DL models.	Restricted assessment on varied IoT datasets and lacks advanced FS mechanisms.

Continued on next page

Authors	Objectives	Findings	Methods	Research gap
Petihakis et al. [19]	To improve an AI-assisted cybersecurity environment to support against adversarial AI threats.	Improved detection performance.	XAI model.	The system focuses on platform integration rather than optimized neural frameworks for attack recognition.
Sheikh et al. [20]	To design a defense approach for learning-based security techniques in cyber-physical systems employing adversarial learning.	Achieved better results.	GAN.	Does not consider hybrid DL approaches or a parameter optimizer for enhanced performance.
Alabsi et al. [21]	To detect DoS and DDoS threats in an IoT platform utilizing a conditional tabular GAN.	Achieved improved detection accuracy.	CGAN technique.	The model concentrates only on particular threats and does not address broader AML attacks.
Dhamija and Bansal [22]	To propose a defensive structure to mitigate multifaceted adversarial threats using attention feature fusion.	Achieved better performance.	Attention mechanism.	This method does not integrate hybrid neural frameworks with optimized feature subset selection.

3. The proposed methodology

This article proposes the HNNF-AMLADRR approach to develop an effective detection framework for adversarial ML threats in intelligent systems operating within 6G and IoT environments, thereby improving robustness against malicious perturbations. Figure 2 illustrates the block diagram of the HNNF-AMLADRR algorithm. As depicted in Figure 2, the framework consists of data preprocessing, feature subset selection, adversarial attack detection, parameter fine-tuning, and experimental evaluation.

- **Step 1: Data preprocessing:** Input data preprocessing is performed using min–max normalization to transform the raw data into a standardized and appropriate format for efficient model training and analysis.
- **Step 2: Feature subset selection:** The ridge regression method is employed for the feature subset selection process to remove irrelevant, noisy, or redundant features, thereby improving the quality of the input data and reducing model complexity.
- **Step 3: Hybrid attack detection:** A hybrid neural network (NN) framework integrating a self-attention-based RBM with an RCNN is utilized. From the given input, the RCNN model captures sequential and spatial characteristics. In contrast, the self-attention technique is used in the RCNN-generated feature maps for identifying vital and related patterns. This self-attention technique enhances the result by allocating more weights to essential features; hence, it improves the detection performance of the attack. This integration enhances the model's capability to accurately identify adversarial attack patterns in intelligent IoT networks.

- **Step 4: Parameter optimization:** The model parameters are fine-tuned using the Adafactor optimizer, which enhances convergence efficiency and improves the overall classification performance of the proposed methodology.
- **Step 5: Experimental assessment:** The experimental evaluation of the HNNF-AMLADRR algorithm is conducted using the AML dataset across three different application scenarios, namely DNS tunneling, platooning, and RUL prediction.

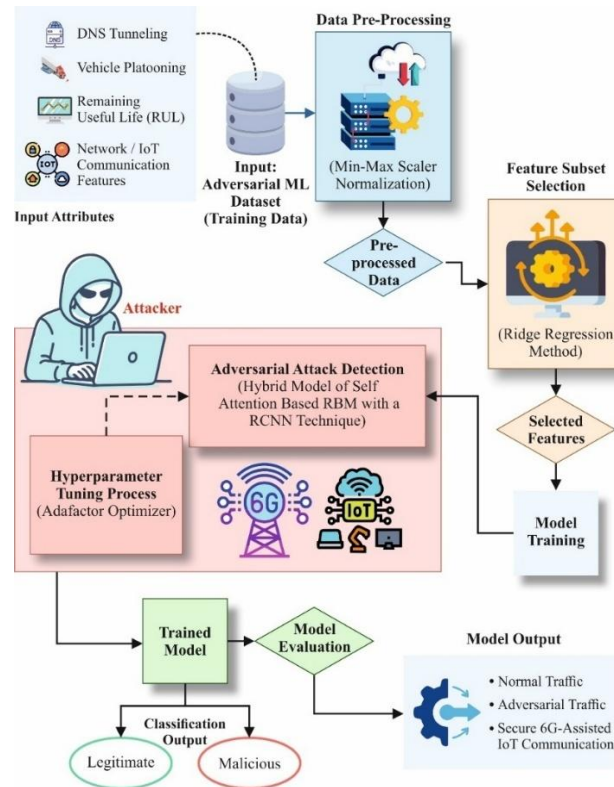


Figure 2. Block diagram of HNNF-AMLADRR model.

3.1. Preprocessing model

Initially, the min-max scaler normalized model can be used to convert the input data into a useful pattern. It effectively normalizes and scales the feature values, ensuring reliability and relevance [23]. The process of normalization ensures that every item of data is standardized. A standardization model named MinMaxScaler increases each value of a feature to a range of zero to one, which improves the stability and performance of the model.

$$l_{std} = \frac{(I - I_{\min})}{(I_{\max} - I_{\min})}, \quad (3.1)$$

$$l_{scaled} = l_{std} * (l_{\max} - l_{\min}) + l_{\min}, \quad (3.2)$$

where I represents the input feature set, and $I_{\min}$ and $I_{\max}$ denote the minimum and maximum values of the input data used in the min-max normalization process. Likewise, l_{std} denotes the normalized intermediate value. The maximal and minimal values are depicted as max and min values in Eqs (3.1) and (3.2).

3.2. Ridge regression-based feature subset selection process

In this section, the ridge regression model is primarily implemented for the feature subset selection procedure to remove redundant or irrelevant features. Ridge regression is used for the reduction of features since it efficiently manages multicollinearity over higher-dimensional IoT features while preserving vital attributes through L2 regularization. Contrary to PCA, which transforms basic features into new elements that reduce interpretability, this method maintains the basic structure of the features. Compared to LASSO, this model eliminates highly correlated features; unlike mutual information, it provides high generalization and consistency for complicated adversarial attack datasets. Feature selection (FS) is vital to optimizing the performance of the model by identifying the most informative features while extracting irrelevant or redundant ones [24]. Various techniques exist for FS, including mutual information, which determines the dependency among variables. However, these models are extensively utilized to choose ridge FS due to its capability to manage multicollinearity by employing L2 regularization. This method ensures that a weaker feature coefficient is penalized, resulting in enhanced generalization and reduced overfitting in high-dimensional data. Ridge regression is a linear regression technique with an L2 regularization term added to the loss function. The major benefits of ridge regression are its ability to penalize large coefficients, thereby ensuring that only the most significant features are retained in the model.

In Eq (3.3), ridge regression alters the typical ordinary least squares (OLS) cost function by adding a term of regularization:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^n \theta_j^2. \quad (3.3)$$

Here, m signifies the training sample count, $J(\theta)$ indicates the cost function of the feature coefficient $\theta = (\theta_1, \theta_2, \dots, \theta_n)$, $\hat{y}_i$ represents the predicted output for the i th training instance, θ_j depicts the coefficient for the j th feature, y_i denotes the actual output for the i th training instance, and λ refers to the regularization parameter, managing the sum of shrinkage implemented to the coefficients.

The first term $\frac{1}{2m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$ depicts the standard mean squared error (MSE). $\lambda \sum_{j=1}^n \theta_j^2$ is an L2 regularization term; here, the addition of squared coefficients is penalized. The regularization term discourages huge values of the coefficient, forcing the method to choose fewer and more significant features, while pushing less relevant features toward 0. The ridge regression solution is attained by solving Eq (3.4):

$$\theta = (Z^T Z + \lambda I)^{-1} Z^T y. \quad (3.4)$$

Here, y denotes the vector of observed values, λI signifies the regularization term, I represents the identity matrix, and Z refers to the matrix of input features.

Equation (3.4) effectively alters the conventional normal equation of linear regression by adding the λI term that stabilizes the matrix inversion, while $Z^T Z$ is near-singular or ill-conditioned. The regularization term precludes overfitting by shrinking the coefficient of features. Ridge regression assures that it retains the most significant features while enhancing efficacy and robustness.

Although ridge regression does not generate sparse coefficients like LASSO, it was used in this architecture for redundancy of features and for handling multicollinearity over highly correlated IoT communication features. The L2 regularization technique balances the weights of features and retains vital correlated attributes, which is helpful in the detection of adverse attacks where various dependent

features contribute to the performance of the classification task. According to the resulting coefficient values, the highly related features were then selected in the final step of the classification process.

3.3. Hybrid adversarial attack detection method

For adversarial attack detection, the presented HNNF-AMLADRR model designs a hybrid model of self-attention-based RBM using the RCNN technique. The self-attention dual classification layers employ an RCNN and RBM. The RBM method is used for efficient representations of data and for the latent feature learning process. The RCNN model is used to capture sequential and spatial attack patterns, where the self-attention technique increases the identification of vital features by allocating adaptive vital weights. Integrating RCNN and RBM provides a unique benefit by employing the unsupervised feature learning ability of RBM with the spatial and sequential processing capability of RCNN. The proposed hybrid network follows a sequential architecture, where the RCNN first captures spatial and sequential attributes from the original data. The learned feature maps are then refined by the self-attention mechanism and passed to the RBM for latent feature learning and final adversarial attack detection.

RCNN: The features of unstructured data are removed to employ the RCNN approach. The hidden layer (HL), input, output, and delay layers contain RCNN. The final HL, $h(t-1)$, serves as feedback to the succeeding HL, $h(t)$, until the following time-step is handled below because of the delay unit. If the input is $x(t)$,

$$h(t) = f(x(t), h(t-1), \theta). \quad (3.5)$$

Now, parameter θ is obtained through the backpropagation method, and f denotes the activation function. Specifically, the recursive and feedforward inputs of the RNN unit are specified by $x(t)$ and $h(t-1)$, respectively. Recursive and feedforward connections were employed to calculate the convolution in a manner that varies from the traditional CNN.

$$h(t)_i = f(w_i^f x(t) + w_i^r h(t-1) + b_i). \quad (3.6)$$

Here, $h(t)_i$ signifies the output of the feature graph, b_i depicts the deviation of the feature graph, and w_i^f and w_i^r refer to weight matrices. Modified features of unstructured textual data are removed utilizing this technology. The output of RCNN is connected to the self-attention layer, whose refined feature representations are subsequently given as input to the RBM for adversarial attack detection. The self-attention layer is a useful DL mechanism that enables a sequence to learn from itself. Thus, self-attention is employed between RCNN and RBM, acting as a weight-sharing tool for interconnection.

Self-attention mechanism: Self-attention is a DL method that allows for finding and highlighting highly related features in the same input by training the relationships in dependency over representations of features. In this framework, the self-attention method generates scores of attention using V (value), K (key), and Q (query) matrices to find the significant features.

The attentive vector for the convolved feature is calculated once, utilizing the attention mechanism on the series of HL obtained from recurrent convolution. This method employs an intra-layer recurrent connection in the convolutional layer. Thus, recurrent relations permit every HL to impact a significant number of adjacent HL.

$$a_i = W_i * [h_i^l, h_i, h_i^r] + b_i. \quad (3.7)$$

Here, h_i depicts the HL, h_i^l and h_i^r refer to the left and right context of HL, and each concealed state is employed as input.

$$C_i = \sum_i \text{Softmax}(a_i). \quad (3.8)$$

While a_i indicates the attention weight, and W_i signifies the weight matrix.

This technique increases contextual feature learning, enhances feature discrimination capacity, and increases the efficiency of detections (adverse attacks) in IoT environments.

RBM: To employ HL of dual stochastic units, $h = (h_1 \dots \dots h_H)$, the RBM joint distribution of input, $x = (x_1 \dots \dots x_D)$, and target class, $y \in \{1, \dots, C\}$ is categorized. To achieve this, describe an energy function.

$$E(y, x, h) = X. \quad (3.9)$$

Now, $X = -h^T W x - b^T x - c^T h - d^T e_y - h^T U e_y$. According to the energy function, the succeeding probability calculates the values of x , y , and h :

$$p(y, x, h) = \frac{\exp(-E(y, x, h))}{Z}. \quad (3.10)$$

Here, Z signifies the normalized constant, ensuring a valid probability distribution. The necessary conditional distributions are straightforward.

$$p(h|y, x) = \prod_j p(h_j|y, x) \text{ with } p(h_j = 1|y, x) = \text{sigm}(c_j + U_{jy} + \sum_i W_{ji} x_i). \quad (3.11)$$

Now, $\text{sigm}(a) = 1/(1 + \exp(-a))$. While conditioning on HL, it is described as

$$p(x|h) = \prod_i p(x_i|h), \text{ with } p(x_i = 1|h) = \text{sigm}(b_i + \sum_j W_{ji} h_j),$$

$$p(y|h) = \frac{\exp(d_y + \sum_i U_{jy} h_i)}{\sum_y^* \exp(d_y + \sum_j U_{jy} h_j)}. \quad (3.12)$$

It is feasible to manage classification by accurately calculating $p(y|h)$. At last, the classifier is effectively able to identify adversarial threats utilizing the RBM. Figure 3 specifies the structure of the RCNN model.

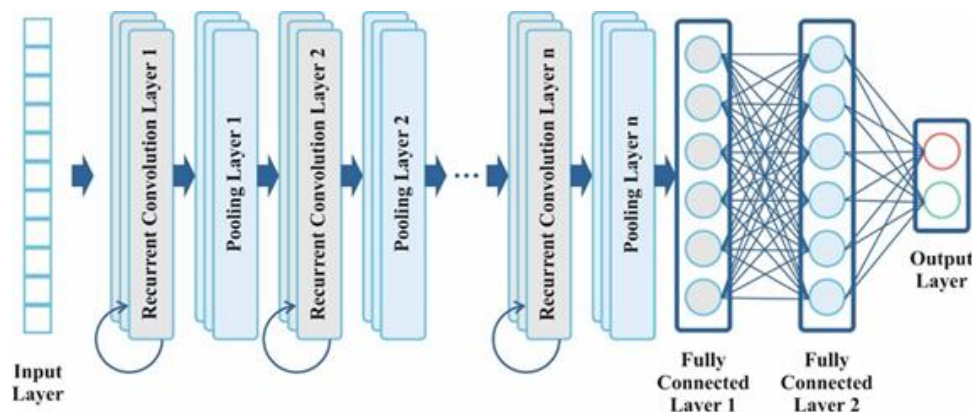


Figure 3. Structure of RCNN model.

3.4. Parameter tuning using the Adafactor model

Finally, parameter fine-tuning has been executed using the Adafactor optimizer to improve the classification solution. The adaptive optimizers require additional memory for storing the variables, namely the first and second moment estimations, as employed in the Adam method, as well as the model weights [25,26]. Adafactor is a form of Adam that is contrasted with alternative methods and is intended to create resource usage more efficiently.

The selection of the Adafactor optimizer is motivated by the technical challenges related to training deep adversarial attack detection models on high-dimensional IoT communication data. Present optimizers like Adam generally demand substantial memory consumption for sustaining second-order parameter statistics that could increase computational overhead in resource-constrained IoT settings. Likewise, Adafactor decreases memory complexity by factored parameterization while maintaining constant and effective gradient updates. This feature makes it more suitable for the presented HNNF-AMLADRR architecture by enhancing convergence efficiency, reducing training overhead, and supporting strong learning for large-scale adversarial feature representations in secure IoT communication networks. Theoretically, unlike Adam, which stores full moment statistics for every parameter, Adafactor decomposes these statistics into low-rank factors, making it more appropriate for large-scale and high-dimensional IoT adversarial learning tasks.

The Adafactor optimizer utilizes several methods to decrease the memory utilization and maximize computational effectiveness. Considering that a subset of the model weights has been calculated by a matrix W , a matrix V of a similar size is required for second-moment estimation of gradients. The mathematical terms $W \in \mathbb{R}^{n \times m}$ and $V \in \mathbb{R}^{n \times m}$ are determined. The second-moment estimation matrix must be estimated as the outer product of column and row vectors $R \in \mathbb{R}^{n \times 1}$ and $C \in \mathbb{R}^{1 \times m}$, correspondingly, $V \approx RC$. From this condition, the matrix V needs the memory size $n + m$ rather than $n \times m$. In the matrix method, the bias-corrected moment estimation matrix with the T^{th} upgrade iteration ($\hat{V}_t$) and the parameters (R_t, C_t) can be denoted by

$$R_t = \hat{\beta}_{2t} R_{t-1} + (1 - \hat{\beta}_{2t})(G_t^2 + \varepsilon_1 1_n 1_m^T) 1_m,$$

$$C_t = \hat{\beta}_{2t} C_{t-1} + (1 - \hat{\beta}_{2t}) 1_n^T (G_t^2 + \varepsilon_1 1_n 1_m^T),$$

$$\hat{V}_t = \frac{R_t C_t}{1_n^T R_t}.$$

Here, the mathematical form T means the transpose operator, and $\hat{\beta}_{2t}$ represents the altered second-decay rate. The symbol ε_1 refers to the first regularization constant. The 1_n and 1_m indicate the column vectors with respect to the size of n and m . In order to decrease the memory usage, the Adafactor removes the first-moment estimation that can be employed in Adam ($\beta_1 = 0$), although this elimination may result in huge updates and training variability. From this condition, the $\hat{\beta}_{2t}$ is described as

$$\hat{\beta}_{2t} = 1 - \frac{1}{\tau^e}, \quad t \geq 1.$$

Here, the mathematical term $e > 0$ remains a scalar that can control the escalation rate. Similarly, the clipped unscaled upgrade $\hat{U}$ and unscaled upgrade U_τ are denoted as

$$U_t = \frac{G_t}{\sqrt{\hat{V}_t}},$$

$$\hat{U}_t = \frac{U_t}{\max(1, \text{RMS}(U_t)/t_c)},$$

where root mean square (RMS) remains the root mean square operator, and t_c refers to the clipping threshold. Hence, if the RMS is exceeded, the term t is used to compute the updates to prevent massive updates.

Simultaneously, the Adafactor dynamically adjusts the rate of learning by calculating the update magnitude relevant to the measurement of the weights.

$$\eta_t = \max(\varepsilon_2, \text{RMS}(w_\tau))q_t.$$

Here, the symbol ε_2 stands for the second regularization constant. Finally, integrating η_t and $\hat{U}$ into the computation model, the update rule of Adafactor is represented by

$$w_{t+1} = w_t - \eta_t \hat{U}_t.$$

The hyperparameters for the Adafactor are used to efficiently perform in large-scale models. Table 2 presents the optimum parameter settings utilized for training the proposed method for efficient performance. The learning rate is fixed at 0.001 and the batch size at 32 to guarantee stable and effective training. The RBM module utilizes 128 hidden units to extract meaningful hidden features, while the RCNN framework deploys 64 convolutional filters with a kernel dimension of 3 and 64 recurrent units to capture spatial and sequential patterns. Additionally, the self-attention mechanism with four heads enhances feature focus. To prevent overfitting, dropout (0.3) and weight decay (0.0001) are executed, and the model is trained for 25 epochs to achieve optimum performance.

Table 2 Parameter table.

Hyperparameter	Description	Optimal values
Learning rate	Step size utilized by the optimizer for weighted updates	0.001
Batch size	No. of training samples processed per iteration	32
RBM hidden units	No. of neurons from the hidden layer of RBM	128
No. of convolution filters	Filters used in RCNN convolution layers	64
Kernel size	Size of convolution filters in RCNN	3
Recurrent units	No. of units in the recurrent component of RCNN	64
Attention heads	No. of heads from the self-attention mechanism	4
Dropout rate	Regularization parameter to prevent overfitting	0.3
Weight decay	Regularization parameter used with the optimizer	0.0001
Training epochs	No. of complete passes by the dataset	25

4. Validation and results

The performance investigation of the HNNF-AMLADRR algorithm is verified under the AML dataset [27]. The AML dataset, accessible on Kaggle, was developed for the evaluation of the strength of ML methods against adversarial attacks across numerous application fields. The dataset

comprises three different settings: Dnstunneling, vehicle platooning, and RUL prediction. The Dnstunneling dataset encompasses network traffic data gathered in Dnstunneling attacks that are commonly utilized to evade network security mechanisms. The platooning dataset contains simulated data from autonomous vehicle platooning infrastructure, where several vehicles travel in coordinated formations. The RUL dataset is concentrated on predictive maintenance by examining aircraft engine data to calculate the residual operational life of modules. All the application scenarios rely on different adversarial ML methods to produce manipulated inputs, thus allowing researchers to analyze the influence of adversarial attacks and develop strong defense mechanisms for protective ML models. The Dnstunneling dataset contains 8998 instances under dual classes. It has 12 features in total, but only 9 are chosen. The platooning dataset has 9000 samples. It contains 6 features, but 5 are chosen. The RUL dataset holds 37,421 samples, with 7 features, but 6 are selected. The proposed HNNF-AMLADRR method was executed in Python 3.11 using TensorFlow and implemented on a system with an Intel Core i7 processor, 16 GB RAM, NVIDIA RTX 3060 GPU, and Windows 11 OS. The dataset was divided into 70% training and 30% testing sets without overlap to avoid data leakage and confirm reliable performance. The complete details of this dataset are given in Table 3.

Table 3. Details of the dataset.

Classes	Dataset		
	Dnstunneling	Platooning	RUL
Normal	2999	3000	16,632
Malicious	5999	6000	20,789
Total	8998	9000	37,421
Number of features	12	6	7
Selected features	9	5	6

Figure 4 represents the classifier analysis of HNNF-AMLADRR on the Dnstunneling dataset. Figures 4(a) and 4(c) demonstrate the confusion matrices with precise recognition and classification of every class of 70% and 30%. Figure 4(b) exhibits the PR inspection, signifying maximal performance in two classes. Figure 4(d) depicts the ROC inspection, exhibiting effective outcomes with a greater value of ROC for individual classes.

Table 4 and Figure 5 display the attack detection outcome of the HNNF-AMLADRR at the Dnstunneling database. Based on 70% TRPE, the HNNF-AMLADRR methodology attains accuracy of 98.32%, precision of 98.13%, recall of 98.06%, F1-score of 98.10%, MCC of 96.19%, and G-measure of 98.10%. At 30% TEPE, the HNNF-AMLADRR attains accuracy of 98.59%, precision of 98.36%, recall of 98.51%, F1-score of 98.44%, MCC of 96.87%, and G-measure of 98.44%.

Figure 6 exemplifies the training (TRAG) accuracy and validation (VALN) accuracy of an HNNF-AMLADRR on the Dnstunneling database over 25 epochs. Primarily, both TRAG and VALN accuracy rise rapidly, reflecting successful identification of patterns within the data. Under each epoch, the VALN accuracy is slightly above the TRAG accuracy, signifying better generalization without overfitting.

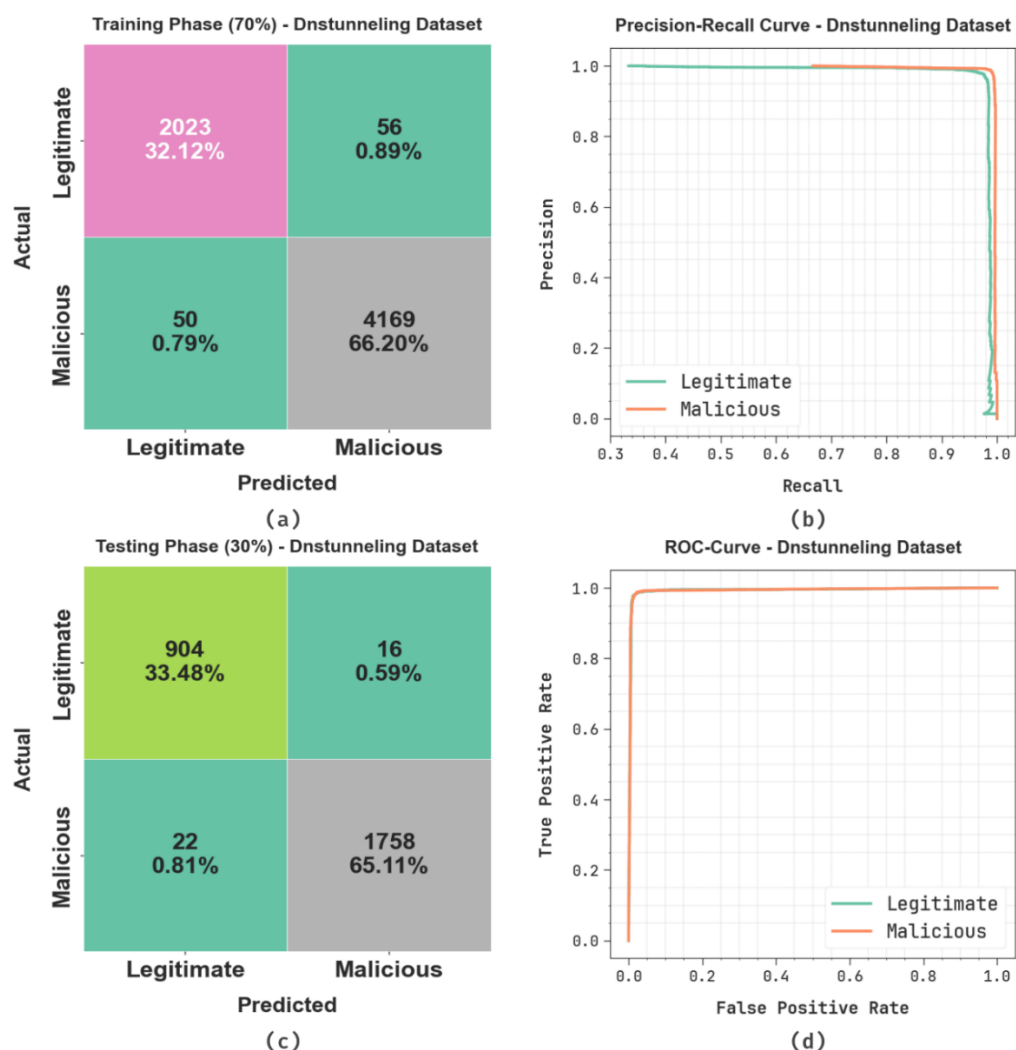


Figure 4. Dnstunneling dataset (a,c) confusion matrices and (b,d) curves of PR and ROC.

Table 4. Attack detection of HNNF-AMLADRR on the Dnstunneling database.

Classes	Accuracy	Precision	Recall	F1-score	MCC	G-measure
Training phase (70%)						
Legitimate	98.32	97.59	97.31	97.45	96.19	97.45
Malicious	98.32	98.67	98.81	98.74	96.19	98.74
Average	98.32	98.13	98.06	98.10	96.19	98.10
Testing phase (30%)						
Legitimate	98.59	97.62	98.26	97.94	96.87	97.94
Malicious	98.59	99.10	98.76	98.93	96.87	98.93
Average	98.59	98.36	98.51	98.44	96.87	98.44

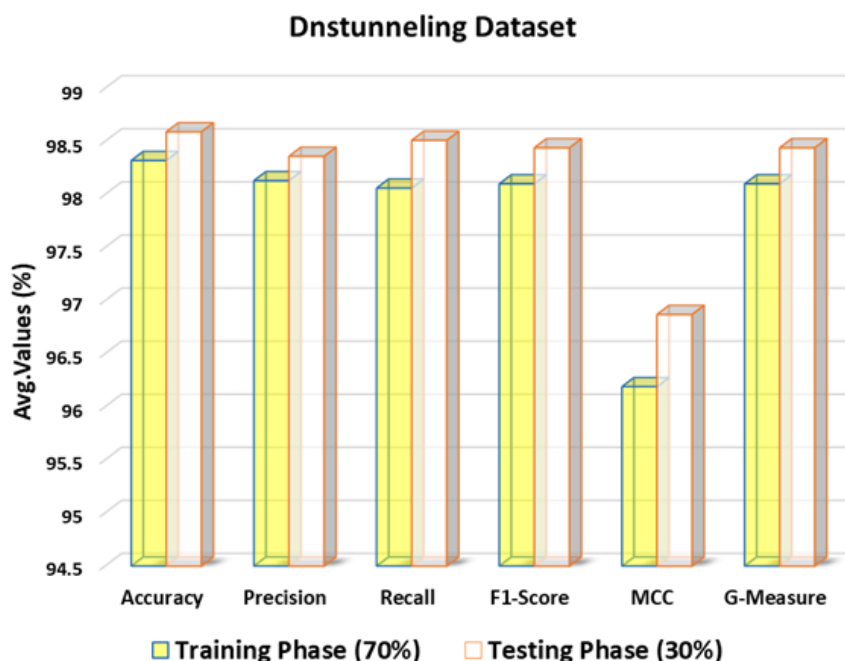


Figure 5. Average of HNNF-AMLADRR on the Dnstunneling database.

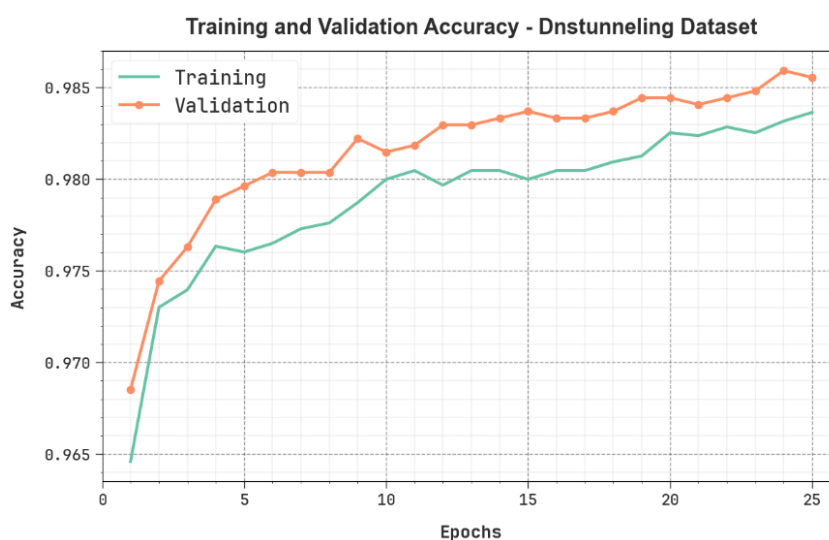


Figure 6. Accuracy graph of HNNF-AMLADRR under the Dnstunneling database.

Figure 7 exemplifies the TRAG and VALN loss of HNNF-AMLADRR on the Dnstunneling database over 25 epochs. First, both TRAG and VALN losses can be observed, indicating that the method starts with an incomplete understanding of the data. As TRAG progresses, both losses are progressively reduced, indicating that the method is proficiently learning and updating its parameters.

Figure 8 portrays the classifier outcome of the HNNF-AMLADRR methodology on the platooning dataset. Figures 8(a) and 8(c) present the confusion matrices with correct classification of each class at 70% and 30%. Figure 8(b) illustrates the PR examination, denoting maximum performance in every class. Finally, Figure 8(d) presents the ROC analysis, representing valuable solutions with superior ROC values for individual classes.

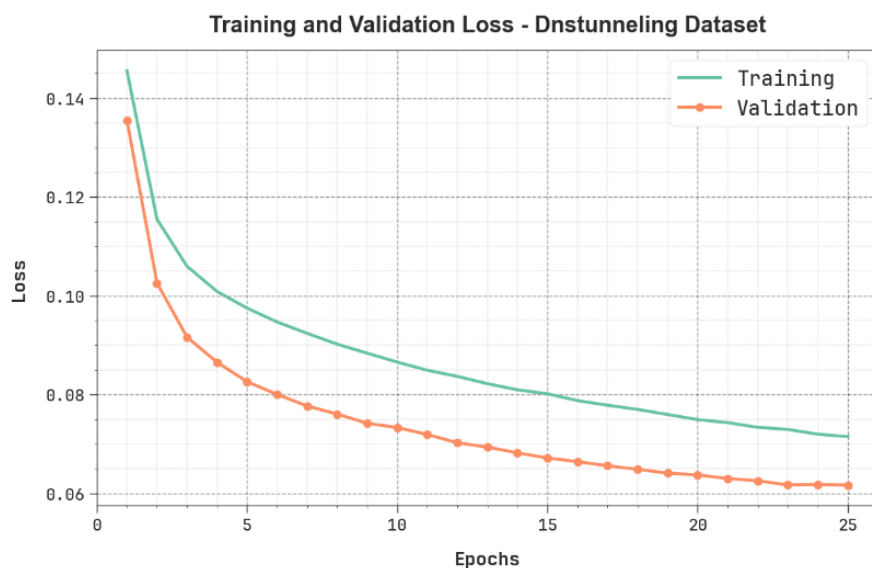


Figure 7. Loss graph of HNNF-AMLADRR on the Dnstunneling database.

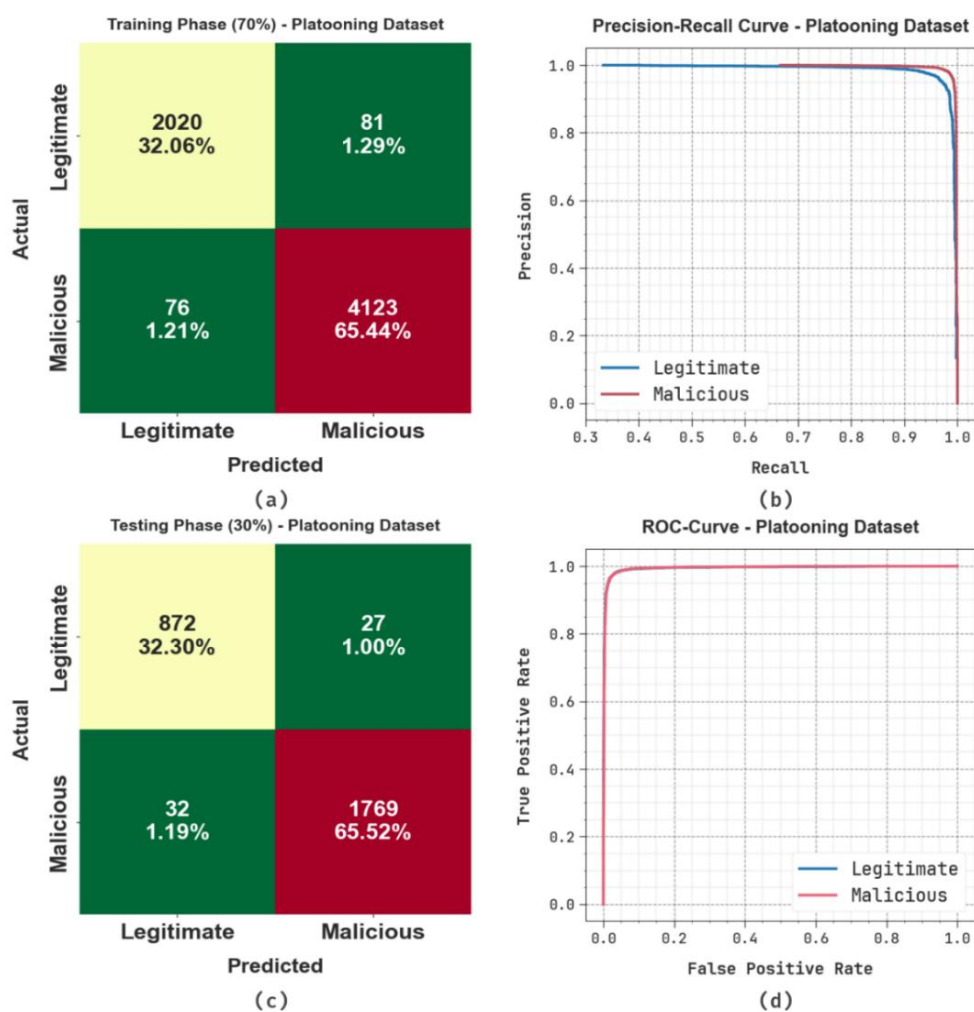


Figure 8. Platooning dataset (a,c) confusion matrices and (b,d) curves of PR and ROC.

Table 5 and Figure 9 signify the attack detection outcome of the HNNF-AMLADRR on the platooning database. Based on 70% TRPE, the HNNF-AMLADRR model attains accuracy, precision, recall, F1-score, MCC, and G-measure of 97.51%, 97.22%, 97.17%, 97.20%, 94.39%, and 97.20%, respectively. At 30% TEPE, the HNNF-AMLADRR reached accuracy, precision, recall, F1-score, MCC, and G-measure of 97.81%, 97.48%, 97.61%, 97.54%, 95.09%, and 97.54%, respectively.

Table 5. Attack detection of the HNNF-AMLADRR model on the platooning dataset.

Classes	Accuracy	Precision	Recall	F1-score	MCC	G-measure
Training phase (70%)						
Legitimate	97.51	96.37	96.14	96.26	94.39	96.26
Malicious	97.51	98.07	98.19	98.13	94.39	98.13
Average	97.51	97.22	97.17	97.20	94.39	97.20
Testing phase (30%)						
Legitimate	97.81	96.46	97.00	96.73	95.09	96.73
Malicious	97.81	98.50	98.22	98.36	95.09	98.36
Average	97.81	97.48	97.61	97.54	95.09	97.54

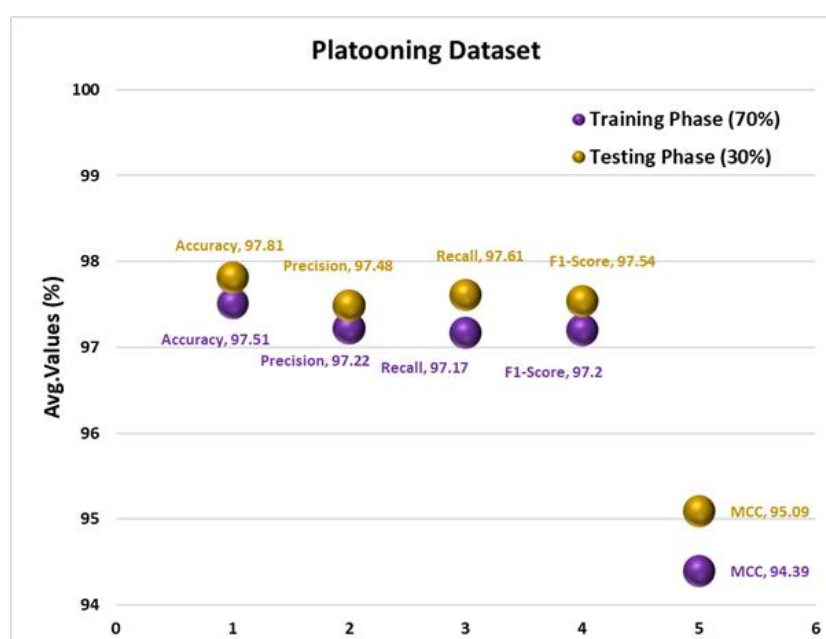


Figure 9. Average of HNNF-AMLADRR with the platooning database.

Figure 10 shows the TRAG accuracy and VALN accuracy of the HNNF-AMLADRR model on the platooning dataset over 25 epochs. At the start, both TRAG and VALN accuracy increase quickly, indicating effective learning of underlying patterns from the data. By the epoch, the VALN accuracy exceeds the TRAG accuracy slightly, implying good generalization without overfitting.

Figure 11 depicts the TRAG and VALN loss of the HNNF-AMLADRR model on the platooning database at 25 epochs. At first, both TRAG and VALN losses can be high, indicating that the model starts with limited consideration of data. As the TRAG process progresses, both losses are reduced, illustrating that the model has successfully learned and enhanced its parameters.



Figure 10. Accuracy graph of HNNF-AMLADRR at the platooning database.



Figure 11. Loss curve of the HNNF-AMLADRR system under the platooning database.

Figure 12 portrays the classifier outcome of the HNNF-AMLADRR methodology under the RUL dataset. Figures 12(a) and 12(c) demonstrate the confusion matrices with accurate detection of every class at 70% and 30%. Figure 12(b) points out the PR outcome, indicating greater performance in two classes. Finally, Figure 12(d) points out the ROC examination, determining effective results with maximal values of ROC for different classes.

Table 6 and Figure 13 depict the attack detection outcome of the HNNF-AMLADRR methodology at the RUL database. Based on 70% TRPE, the HNNF-AMLADRR reached accuracy, precision, recall, F1-score, MCC, and G-measure of 95.80%, 95.77%, 95.73%, 95.75%, 91.50%, and 95.75%, respectively. At 30% TEPE, the HNNF-AMLADRR approach accomplished accuracy, precision, recall, F1-score, MCC, and G-measure of 96.41%, 96.37%, 96.36%, 96.36%, 92.73%, and 96.36%, respectively.

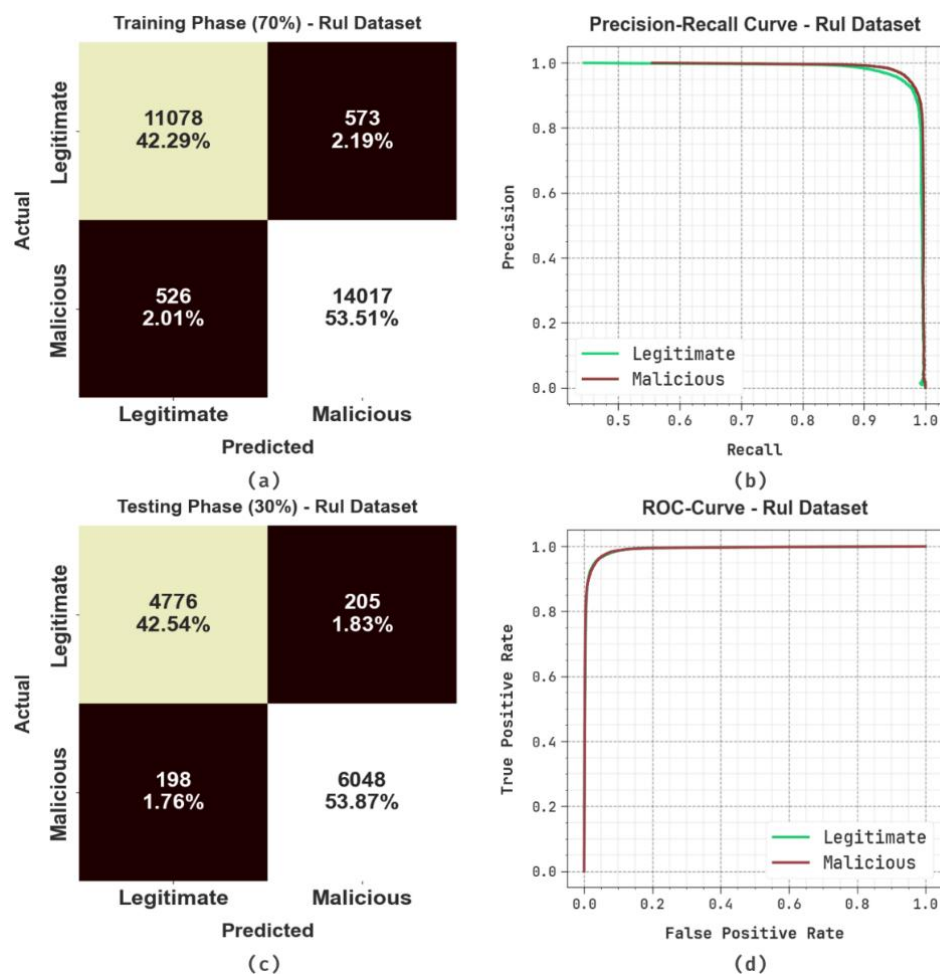


Figure 12. RUL dataset (a,c) confusion matrices and (b,d) curves of PR and ROC.

Table 6. Attack recognition of HNNF-AMLADRR on the RUL database.

Classes	Accuracy	Precision	Recall	F1-score	MCC	G-measure
Training phase (70%)						
Legitimate	95.80	95.47	95.08	95.27	91.50	95.27
Malicious	95.80	96.07	96.38	96.23	91.50	96.23
Average	95.80	95.77	95.73	95.75	91.50	95.75
Testing phase (30%)						
Legitimate	96.41	96.02	95.88	95.95	92.73	95.95
Malicious	96.41	96.72	96.83	96.78	92.73	96.78
Average	96.41	96.37	96.36	96.36	92.73	96.36

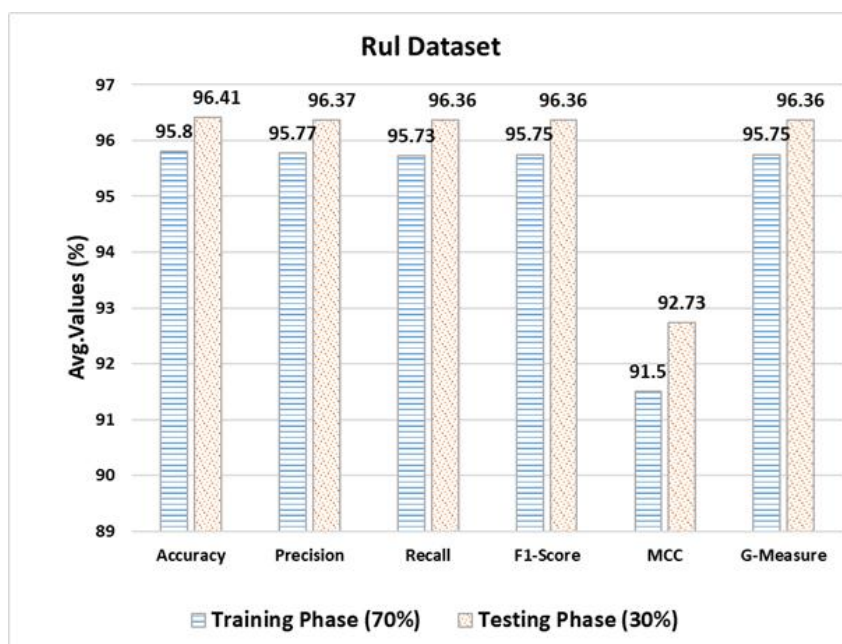


Figure 13. Average of HNNF-AMLADRR on the RUL database.

Figure 14 demonstrates the TRAG accuracy and VALN accuracy of the HNNF-AMLADRR on the RUL database at 25 epochs. Primarily, both TRAG and VALN accuracies rise rapidly, denoting efficient learning of patterns from the data. Over the epochs, the VALN accuracy is slightly above the TRAG accuracy, demonstrating optimal generalization without overfitting.

Figure 15 illustrates the TRAG and VALN loss of the HNNF-AMLADRR on the RUL database at 25 epochs. Initially, both TRAG and VALN losses can be significant, indicating that the approach starts with a limited understanding of the data. As TRAG progresses, both losses steadily decrease, indicating that the method is effectively learning and updating its parameters.

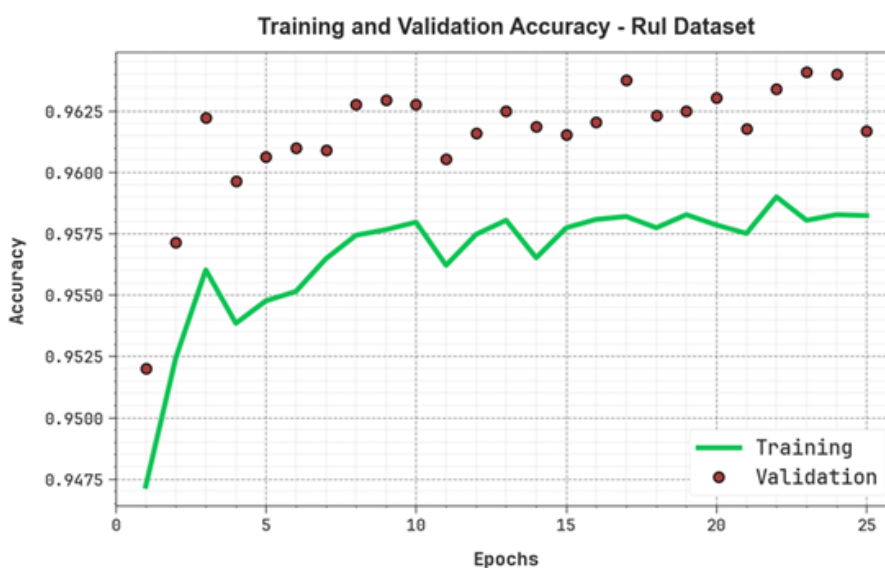


Figure 14. Accuracy graph of HNNF-AMLADRR on the RUL database.

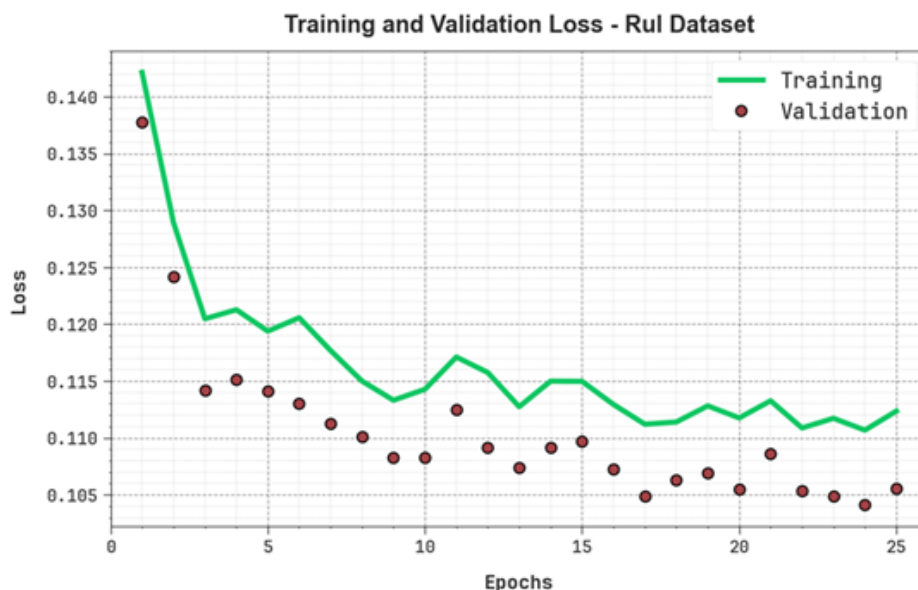


Figure 15. Loss curve of HNNF-AMLADRR on the RUL database.

Table 7 describes the comparative investigation of the HNNF-AMLADRR system on Dnstunneling, platooning, and RUL databases with existing methods [28–30]. Figure 16 presents the comparison analysis of the HNNF-AMLADRR approach on the Dnstunneling dataset. The HNNF-AMLADRR algorithm achieved higher results, with an accuracy, precision, recall, and F1-score of 98.59%, 98.36%, 98.51%, and 98.44%, respectively. Existing systems, namely LSTM, GRU, VAE, Vgg19, ViT, ResNet50, GAN+CNN-LSTM, RNN-BiLSTM-CRF, SDAE-SVM, decision tree, KNN, SVM, gradient boost, MLPNN, random forest, and SGD, demonstrated inferior performance.

Figure 17 showcases the comparative outcome of the HNNF-AMLADRR technique on the platooning dataset. The figure indicates that the HNNF-AMLADRR outperformed the other methods. Regarding accuracy, the HNNF-AMLADRR model achieved a top accuracy of 97.81%, whereas current methodologies—namely LSTM, GRU, VAE, Vgg19, ViT, ResNet50, GAN+CNN-LSTM, RNN-BiLSTM-CRF, SDAE-SVM, decision tree, KNN, SVM, gradient boost, MLPNN, random forest, and SGD—attained lower accuracies of 93.08%, 90.50%, 92.51%, 92.48%, 90.04%, 89.57%, 89.04%, 92.56%, 91.26%, 95.97%, 91.46%, 89.76%, 92.20%, 91.19%, 89.57%, and 92.36%, respectively. Lastly, in terms of F1-score, the IWECHM-FRHAM approach achieved a maximum F1-score of 97.54%, while the LSTM, GRU, VAE, Vgg19, ViT, ResNet50, GAN+CNN-LSTM, RNN-BiLSTM-CRF, SDAE-SVM, SVM, Naïve Bayes, fast trees, ResNet-50, Inception V3, AdaBoost, and decision tree methodologies recorded a minimum F1-score of 88.15%, 90.69%, 90.84%, 89.36%, 91.71%, 90.09%, 90.62%, 87.73%, 91.81%, 91.01%, 89.64%, 97.43%, 97.40%, 93.16%, 90.44%, and 90.16%, respectively.

Table 7 Comparative outcome of HNNF-AMLADRR under Dnstunneling, platooning, and RUL databases.

Classifiers	Dnstunneling				Platooning				RUL			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
LSTM	90.09	87.52	86.44	86.98	93.08	88.15	88.14	88.15	87.63	88.39	92.48	90.39
GRU	90.03	86.50	87.51	87.00	90.50	93.98	87.63	90.69	92.97	90.13	88.83	89.48
VAE	91.60	92.83	93.88	93.35	92.51	88.78	92.99	90.84	88.04	94.85	86.46	90.46
Vgg19	89.45	89.72	94.96	92.27	92.48	87.47	91.33	89.36	93.56	93.58	94.18	93.88
ViT	91.66	88.80	87.11	87.95	90.04	90.95	92.48	91.71	90.83	86.61	86.39	86.50
ResNet50	90.54	94.37	93.43	93.90	89.57	92.72	87.61	90.09	87.86	87.42	93.15	90.19
GAN+CNN-LSTM	91.77	93.32	92.71	92.93	89.04	86.75	90.94	90.62	94.18	86.15	91.87	92.66
RNN-BiLSTM-CRF	86.62	86.31	88.15	92.27	92.56	94.17	93.53	87.73	90.09	85.08	93.53	88.67
SDAE-SVM	85.26	86.68	94.88	85.57	91.26	91.69	86.49	91.81	94.35	85.84	93.94	92.01
Decision tree	98.03	91.89	97.52	89.97	95.97	94.38	94.58	91.01	90.15	89.55	89.51	95.85
KNN	89.75	90.88	92.36	97.42	91.46	95.51	96.37	89.64	93.48	92.85	95.70	89.40
SVM	92.14	92.60	90.89	94.34	89.76	90.51	93.27	97.43	92.15	95.16	95.38	94.77
Gradient boost	92.19	96.30	96.89	97.19	92.20	90.48	96.47	97.40	90.15	93.94	89.32	92.32
MLPNN	93.36	91.43	95.65	92.50	91.19	92.64	96.40	93.16	94.60	90.80	95.30	96.15
Random forest	95.96	90.74	90.12	90.59	89.57	95.13	94.24	90.44	94.38	92.61	95.76	95.23
SGD	96.95	90.43	96.30	90.46	92.36	89.91	89.54	90.16	91.67	94.14	92.12	95.31
HNNF-AMLADRR	98.59	98.36	98.51	98.44	97.81	97.48	97.61	97.54	96.41	96.37	96.36	96.36

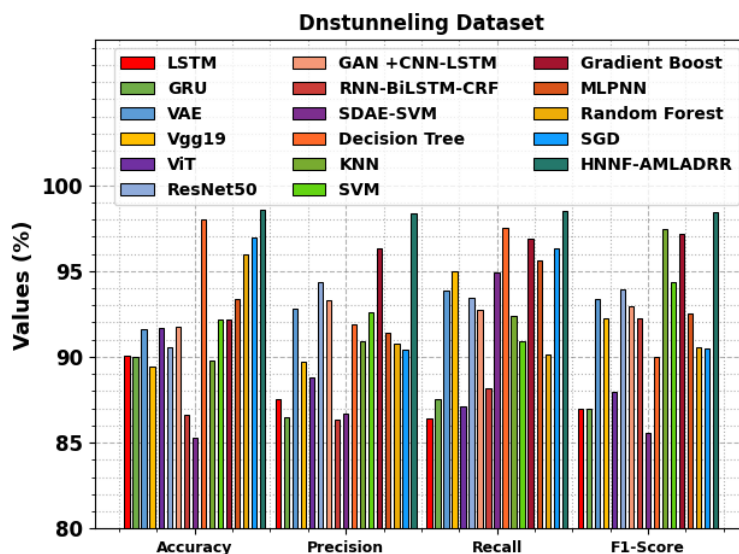


Figure 16. Comparative outcome of HNNF-AMLADRR under the Dnstunneling database.

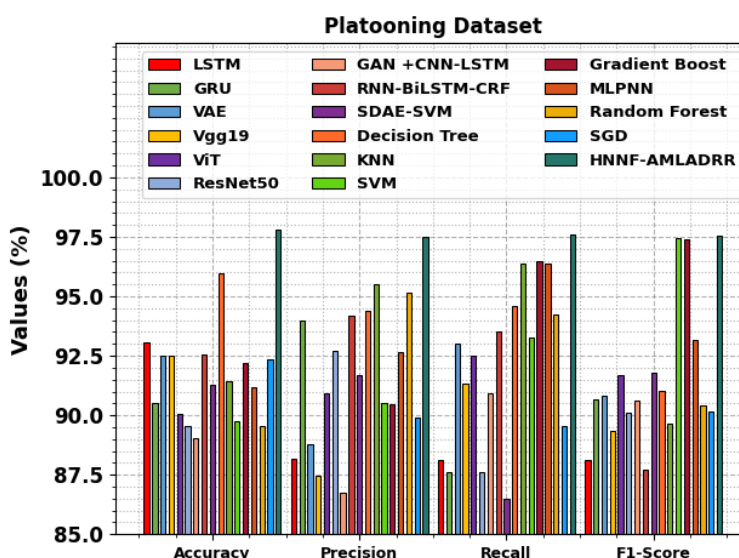


Figure 17. Comparative outcome of HNNF-AMLADRR on the platooning database.

Figure 18 portrays the comparative study of the HNNF-AMLADRR methodology at the RUL dataset. The LSTM, GRU, VAE, Vgg19, ViT, ResNet50, RNN-BiLSTM-CRF, decision tree, gradient boost, and SGD methodologies achieved lower outcomes, with accuracies of 87.63%, 92.97%, 88.04%, 93.56%, 90.83%, 87.86%, 90.09%, 90.15%, 90.15%, and 91.67%, respectively. Additionally, the KNN, SVM, MLPNN, GAN+CNN-LSTM, SDAE-SVM, and random forest obtained slightly better outcomes with accuracies of 93.48%, 92.15%, 94.60%, 94.18%, 94.35%, and 94.38%, while the HNNF-AMLADRR algorithm achieved the highest solution with accuracy, precision, recall, and F1-score of 96.41%, 96.37%, 96.36%, and 96.36%, respectively.

In Table 8, the error analysis of the proposed HNNF-AMLADRR system compared with existing methods under the three datasets is depicted. The LSTM, GRU, VAE, Vgg19, ViT, ResNet50, SDAE-SVM, RNN-BiLSTM-CRF, and KNN approaches demonstrated the worst performance, while the

GAN+CNN-LSTM, SVM, gradient boost, and MLPNN systems exhibited slightly better results. Meanwhile, the random forest, SGD, and DT methodologies showed reasonable performance. However, the proposed HNNF-AMLADRR algorithm outperformed the other methods, with lower errors of 1.41%, 2.19%, and 3.59% across the three datasets.

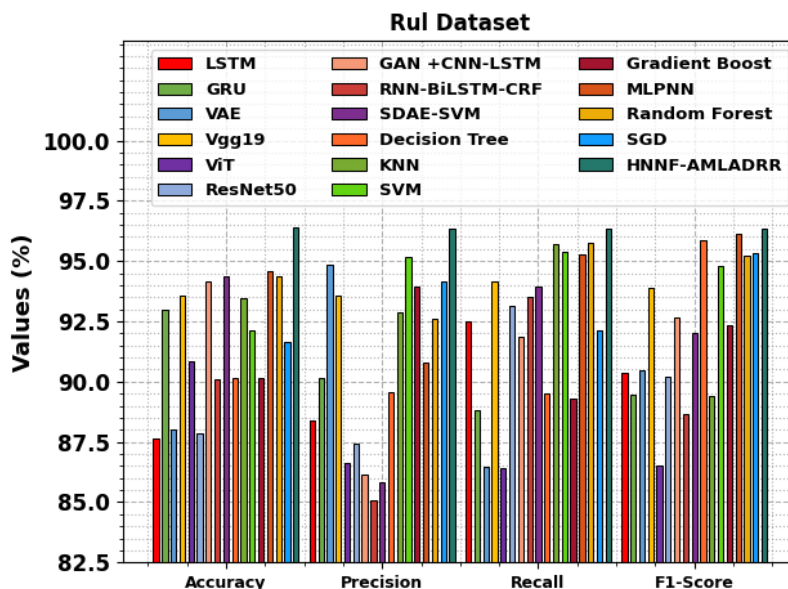


Figure 18. Comparative outcome of HNNF-AMLADRR with the RUL database.

Table 8. Error and CT analysis of the HNNF-AMLADRR algorithm compared with other methods under three datasets.

Classifiers	Error analysis (%)			Computational time (s)		
	Dnstunneling	Platooning	RUL	Dnstunneling	Platooning	RUL
LSTM	9.91	6.92	12.37	4.84	7.06	7.45
GRU	9.97	9.50	7.03	9.60	8.92	5.98
VAE	8.40	7.49	11.96	8.52	11.24	5.85
Vgg19	10.55	7.52	6.44	8.90	10.56	8.64
ViT	8.34	9.96	9.17	9.68	5.37	8.53
ResNet50	9.46	10.43	12.14	11.73	8.70	5.88
GAN+CNN-LSTM	8.23	10.96	5.82	8.87	8.49	5.43
RNN-BiLSTM-CRF	13.38	7.44	9.91	4.79	6.87	4.71
SDAE-SVM	14.74	8.74	5.65	6.01	8.17	8.70
Decision tree	1.97	4.03	9.85	16.09	18.68	8.05
KNN	10.25	8.54	6.52	9.40	5.63	7.98
SVM	7.86	10.24	7.85	17.76	4.71	14.29
Gradient boost	7.81	7.80	9.85	4.80	5.32	11.59
MLPNN	6.64	8.81	5.40	5.66	14.75	14.23
Random forest	4.04	10.43	5.62	16.67	15.88	3.57
SGD	3.05	7.64	8.33	19.52	5.96	10.51
HNNF-AMLADRR	1.41	2.19	3.59	3.01	2.67	1.99

Figure 19 demonstrates the computational time (CT) outcome of the HNNF-AMLADRR system under three datasets. Based on the Dnstunneling dataset, the HNNF-AMLADRR approach attained a lower CT of 3.01 s while the LSTM, GRU, VAE, Vgg19, ViT, ResNet50, GAN+CNN-LSTM, RNN-BiLSTM-CRF, SDAE-SVM, DT, KNN, SVM, Gradient Boost, MLPNN, RF, and SGD systems gained a higher CT of 4.84, 9.60, 8.52, 8.90, 9.68, 11.73, 8.87, 4.79, 6.01, 16.09, 9.40, 17.76, 4.80, 5.66, 16.67, and 19.52 s. In addition, based on the platooning dataset, the HNNF-AMLADRR approach reached a lower CT of 2.67 s, while the LSTM, GRU, VAE, Vgg19, ViT, ResNet50, GAN+CNN-LSTM, RNN-BiLSTM-CRF, SDAE-SVM, DT, KNN, SVM, gradient boost, MLPNN, RF, and SGD models obtained greater CTs of 7.06, 8.92, 11.24, 10.56, 5.37, 8.70, 8.49, 6.87, 8.17, 18.68, 5.63, 4.71, 5.32, 14.75, 15.88, and 5.96 s. Finally, based on the RUL dataset, the HNNF-AMLADRR approach accomplished a minimum CT of 1.99 s, while the LSTM, GRU, VAE, Vgg19, ViT, ResNet50, GAN+CNN-LSTM, RNN-BiLSTM-CRF, SDAE-SVM, DT, KNN, SVM, gradient boost, MLPNN, RF, and SGD approaches reached maximum CTs of 7.45, 5.98, 5.85, 8.64, 8.53, 5.88, 5.43, 4.71, 8.70, 8.05, 7.98, 14.29, 11.59, 14.23, 3.57, and 10.51 s.

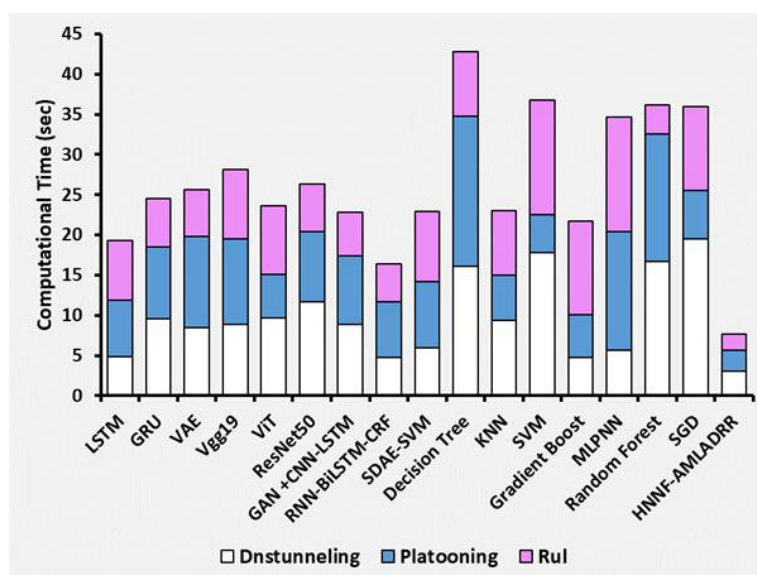


Figure 19. CT outcome of the HNNF-AMLADRR system under three datasets.

Table 9 indicates the ablation evaluation of diverse components in the HNNF-AMLADRR model under three datasets. First, on the Dnstunneling dataset, without ridge regression, the model achieved an accuracy of 95.94%, a precision of 96.03%, a recall of 95.95%, and an F1-score of 95.96%. Also, without a self-attention RBM system, it demonstrated a slightly higher accuracy of 96.72%, a precision of 96.65%, a recall of 96.65%, and an F1-score of 96.58%. Without the RCNN technique, it achieved a slightly higher accuracy of 97.31%, a precision of 97.25%, a recall of 97.26%, and an F1-score of 97.16%. Meanwhile, without Adafactor, optimizer algorithms exhibited closer and more moderate performances. Finally, the HNNF-AMLADRR (ridge regression+self-attention RBM+RCNN+Adafactor) system achieved the highest outputs, with an accuracy of 98.59%, a precision of 98.36%, a recall of 98.51%, and an F1-score of 98.44%, emphasizing the efficiency of the components.

Table 9. Ablation study of HNNF-AMLADRR algorithm under three datasets.

Classifiers	Dnstunneling				Platooning				RUL			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Without ridge regression	95.94	96.03	95.95	95.96	95.21	95.01	95.08	95.19	93.64	93.66	94.23	93.70
Without self-attention RBM	96.72	96.65	96.65	96.58	95.91	95.62	95.73	95.87	94.20	94.37	94.81	94.33
Without RCNN	97.31	97.25	97.26	97.16	96.42	96.15	96.28	96.50	94.98	94.99	95.32	95.00
Without Adafactor optimizer	97.88	97.82	97.96	97.73	97.15	96.89	96.84	97.01	95.75	95.64	95.82	95.65
HNNF-AMLADRR (Ridge regression+self-attention RBM+RCNN+Adafactor)	98.59	98.36	98.51	98.44	97.81	97.48	97.61	97.54	96.41	96.37	96.36	96.36

Second, in the platooning dataset, without ridge regression, the model achieved an accuracy of 95.21%, a precision of 95.01%, a recall of 95.08%, and an F1-score of 95.19%. Also, without self-attention, the RBM system demonstrated slightly higher accuracy of 95.91%, precision of 95.62%, recall of 95.73%, and F1-score of 95.87%. Without the RCNN technique, it obtained a slightly higher accuracy of 96.42%, precision of 96.15%, recall of 96.28%, and F1-score of 96.50%. Meanwhile, without Adafactor, optimizer algorithms exhibited closer and moderate performances. Finally, the HNNF-AMLADRR (ridge regression+self-attention RBM+RCNN+Adafactor) system achieved the highest outputs with an accuracy of 97.81%, precision of 97.48%, recall of 97.61%, and F1-score of 97.54%, emphasizing the efficiency of the components.

At last, with the RUL dataset, without ridge regression, the model achieved an accuracy of 93.64%, a precision of 93.66%, a recall of 94.23%, and an F1-score of 93.70%. Also, the self-attention RBM system demonstrated a slightly higher accuracy of 94.20%, precision of 94.37%, recall of 94.81%, and F1-score of 94.33%. Without the RCNN technique, it illustrated a slightly higher accuracy of 94.98%, precision of 94.99%, recall of 95.32%, and F1-score of 95.00%. Meanwhile, without the Adafactor optimizer, algorithms exhibited closer and moderate performances. Finally, the HNNF-AMLADRR (ridge regression+self-attention RBM+RCNN+Adafactor) system achieved the highest outputs, with an accuracy of 96.41%, precision of 96.37%, recall of 96.36%, and F1-score of 96.36%, emphasizing the efficiency of the components.

The proposed adversarial attack detection framework requires stable and discriminative feature learning from highly correlated IoT network traffic data. Ridge regression is more suitable for this task because adversarial communication patterns often contain interdependent features that contribute collectively to attack detection performance. By reducing coefficient variance without completely removing features, ridge regression improves model robustness, minimizes overfitting, and supports consistent feature representation for the subsequent self-attention RCNN classifier. This characteristic makes it more effective for secure IoT communication environments compared with PCA, LASSO, and mutual information-based feature selection approaches.

5. Conclusions

In this manuscript, we have introduced the HNNF-AMLADRR system for effective detection of AML attacks, thereby improving the security of intelligent systems in 6G and IoT environments against malicious perturbations. First, the min-max scaler normalization model is employed in the data preprocessing phase. Furthermore, the ridge regression model is mainly implemented for the feature subset selection procedure to remove redundant or irrelevant features. For adversarial attack detection, the presented HNNF-AMLADRR model designs a hybrid model of self-attention-based RBM using the RCNN technique. Finally, parameter tuning is carried out using the Adafactor optimizer to improve the classification performance. The empirical evaluation of the HNNF-AMLADRR method is conducted using the AML dataset across three different applications, namely DNS tunneling, platooning, and RUL. The simulation results indicate the superior performance of the HNNF-AMLADRR methodology, achieving accuracy of 98.59%, 97.81%, and 96.41%, respectively, outperforming existing approaches.

The presented HNNF-AMLADRR architecture is suitable for several emerging 6G-enabled IoT conditions where secure and reliable communication is critical. In smart healthcare systems, it could play an important role, such as identifying adversarial manipulations in patient monitoring and medical data transmission. In autonomous and connected vehicles, the model supports secure vehicular communication by detecting malicious traffic and assuring data integrity. It is also applicable to smart city networks and industrial IoT conditions, where large-scale sensor networks need strong protection against adversarial attacks. Overall, the framework provides a scalable and effective solution for improving security, reliability, and trustworthiness in next-generation 6G-IoT communication systems.

Author contributions

M. A. AlAqil and A. K. Alkhalifa: Conceptualization; M. A. AlAqil, S. Alahmari and N. NEMRI: Data curation, Formal analysis; M. A. AlAqil, A. K. Alkhalifa, N. NEMRI, S. Alshahrani, S. Almutairi, A. Ben Miled: Funding support, Investigation, Methodology; S. Almutairi: Project administration, Resources; M. A. AlAqil, A. K. Alkhalifa, S. Alahmari, N. NEMRI, S. Alshahrani, S. Alamro, S. Almutairi, A. Ben Miled: Discussion, Writing–review and editing, Writing–original draft; S. Alamro: Software, Supervision, Validation, Visualisation. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Data availability statement

The data that support the findings of this study are openly available in the Kaggle repository at <https://www.kaggle.com/datasets/cnriciit/adversarial-machine-learning-dataset> [27].

Funding

The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through Large Research Project under grant number RGP2/320/46. This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia, under project Grant KFU264226. Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R721), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would like to thank the Deanship of Scientific Research at Shaqra University for supporting this work. The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number “NBU-FPEJ-2026-2847-02”. The authors are thankful to the Deanship of Graduate Studies and Scientific Research at University of Bisha for supporting this work through the Fast-Track Research Support Program.

Conflict of interest

The authors declare no conflicts of interest.

References

1. R. Kumar, J. Dutta, N. Vamsi, U. S. Varri, D. Puthal, Next-generation security in the 6G era: The role of AI in safeguarding future networks, *IEEE Access*, **14** (2026), 17347–17380. <https://doi.org/10.1109/ACCESS.2025.3650208>
2. A. Tripathi, A. S. Anagha, A. S. Kumar, S. Anjankar, S. Balpande, S. W. Prakash, Enhancing cybersecurity resilience in 6G networks using adversarial machine learning, In: *Emerging perspectives and applications of computational intelligence and smart systems*, CRC Press, 2025, 338–344.
3. R. Huang, Y. C. Li, Adversarial attack mitigation strategy for machine learning-based network attack detection model in power system, *IEEE Trans. Smart Grid*, **14** (2023), 2367–2376. <https://doi.org/10.1109/TSG.2022.3217060>
4. S. Chen, M. H. Xue, L. L. Fan, S. Hao, L. H. Xu, H. J. Zhu, et al., Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach, *Comput. Secur.*, **73** (2018), 326–344. <https://doi.org/10.1016/j.cose.2017.11.007>
5. V. R. Kebande, S. Alawadi, F. M. Awaysheh, J. A. Persson, Active machine learning adversarial attack detection in the user feedback process, *IEEE Access*, **9** (2021), 36908–36923. <https://doi.org/10.1109/ACCESS.2021.3063002>
6. Y. L. Wang, T. Sun, S. H. Li, X. Yuan, W. Ni, E. Hossain, et al., Adversarial attacks and defenses in machine learning-empowered communication systems and networks: A contemporary survey, *IEEE Commun. Surv. Tutor.*, **25** (2023), 2245–2298. <https://doi.org/10.1109/COMST.2023.3319492>
7. F. Mumcu, K. Doshi, Y. Yilmaz, Adversarial machine learning attacks against video anomaly detection systems, In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2022, 205–212. <https://doi.org/10.1109/CVPRW56347.2022.00034>

8. M. Standen, J. Kim, C. Szabo, Adversarial machine learning attacks and defences in multi-agent reinforcement learning, *ACM Comput. Surv.*, **57** (2025), 1–35. <https://doi.org/10.1145/3708320>
9. A. Sharma, S. Rani, Enhancing 6G-IoT network security: A trustworthy and responsible AI-driven stacked-hybrid model for attack detection, *IEEE Int. Things J.*, **13** (2026), 7777–7784. <https://doi.org/10.1109/JIOT.2025.3566403>
10. H. Y. Lin, B. Biggio, Adversarial machine learning: attacks from laboratories to the real world, *Computer*, **54** (2021), 56–60. <https://doi.org/10.1109/MC.2021.3057686>
11. F. O. Catak, S. Y. Yayilgan, Deep neural network based malicious network activity detection under adversarial machine learning attacks, In: *International Conference on Intelligent Technologies and Applications*, Cham: Springer, 2021, 280–291. https://doi.org/10.1007/978-3-030-71711-7_23
12. A. V. Ribeiro, A. L. R. Madureira, L. N. Sampaio, Secure by design: Merging network and security bootstrapping for IoT systems through NDN, *Comput. Netw.*, **280** (2026), 112162. <https://doi.org/10.1016/j.comnet.2026.112162>
13. H. Shen, J. C. Wu, J. Wang, Z. G. Wu, Adversarial dynamic games for Markov jump systems: A policy iteration Q-learning method, *Automatica*, **183** (2026), 112591. <https://doi.org/10.1016/j.automatica.2025.112591>
14. H. Shen, C. J. Peng, H. C. Yan, S. Y. Xu, Data-driven near optimization for fast sampling singularly perturbed systems, *IEEE Trans. Automat. Control*, **69** (2024), 4689–4694. <https://doi.org/10.1109/TAC.2024.3352703>
15. H. Ke, J. Xu, Y. Wang, H. Y. Chen, Z. P. Shen, Adversarial machine learning in cybersecurity: Attacks and defenses, *Int. J. Manag. Sci. Res.*, **8** (2025), 26–33. [https://doi.org/10.53469/ijomsr.2025.08\(02\).04](https://doi.org/10.53469/ijomsr.2025.08(02).04)
16. S. R. Bommana, S. Veeramachaneni, S. E. Ahmed, M. B. Srinivas, Addressing adversarial attacks in IoT using deep learning AI models, *IEEE Access*, **13** (2025), 50437–50449. <https://doi.org/10.1109/ACCESS.2025.3552529>
17. K. Barik, S. Misra, L. Fernandez-Sanz, Adversarial attack detection framework based on optimized weighted conditional stepwise adversarial network, *Int. J. Inform. Secur.*, **23** (2024), 2353–2376. <https://doi.org/10.1007/s10207-024-00844-w>
18. X. W. Yuan, S. Han, W. Huang, H. L. Ye, X. L. Kong, F. Zhang, A simple framework to enhance the adversarial robustness of deep learning-based intrusion detection system, *Comput. Secur.*, **137** (2024), 103644. <https://doi.org/10.1016/j.cose.2023.103644>
19. G. Petihakis, A. Farao, P. Bountakas, A. Sabazioti, J. Polley, C. Xenakis, AIAS: AI-assisted cybersecurity platform to defend against adversarial AI attacks, In: *ARES '24: Proceedings of the 19th International Conference on Availability, Reliability Security*, 2024, 1–7. <https://doi.org/10.1145/3664476.3669920>
20. Z. A. Sheikh, Y. Singh, P. K. Singh, P. J. S. Gonçalves, Defending the defender: Adversarial learning based defending strategy for learning based security methods in cyber-physical systems (CPS), *Sensors*, **23** (2023), 5459. <https://doi.org/10.3390/s23125459>
21. B. A. Alabsi, M. Anbar, S. D. A. Rihan, Conditional tabular generative adversarial based intrusion detection system for detecting DDoS and DoS attacks on the Internet of Things networks, *Sensors*, **23** (2023), 5644. <https://doi.org/10.3390/s23125644>

22. L. Dhamija, U. Bansal, AFLF: A defensive framework to defeat multi-faceted adversarial attacks via attention feature fusion, *Evolv. Syst.*, **16** (2025), 20. <https://doi.org/10.1007/s12530-024-09643-z>
23. B. R. Cherukuri, Advanced multi class cyber security attack classification in IoT based wireless sensor networks using context aware depthwise separable convolutional neural network, *J. Mach. Comput.*, **5** (2025), 814–830. <https://doi.org/10.53759/7669/jmc202505064>
24. C. S. Kodete, K. B. Raju, K. Karmakonda, S. Sikindar, J. V. N. Ramesh, N. S. K. M. K. Tirumanadham, Optimizing intrusion detection with triple boost ensemble for enhanced detection of rare and evolving network attacks, *Int. J. Electr. Electron. Eng. Telecommun.*, **14** (2025), 115–129. <https://doi.org/10.18178/ijeetc.14.3.115-129>
25. Y. F. Zhang, X. Z. Gao, X. Peng, J. Q. Ye, X. Li, Attention-based recurrent temporal restricted Boltzmann machine for radar high resolution range profile sequence recognition, *Sensors*, **18** (2018), 1585. <https://doi.org/10.3390/s18051585>
26. D. Altinel, Development of deep learning optimizers: Approaches, concepts, and update rules, 2025, arXiv: 2509.18396.
27. CNR-IEIIT, *Adversarial Machine Learning Dataset*, Kaggle, 2022. Available from: <https://www.kaggle.com/datasets/cnrieiit/adversarial-machine-learning-dataset>.
28. I. Vaccari, A. Carlevaro, S. Narteni, E. Cambiaso, M. Mongelli, eXplainable and reliable against adversarial machine learning in data analytics, *IEEE Access*, **10** (2022), 83949–83970. <https://doi.org/10.1109/ACCESS.2022.3197299>
29. A. S. Albahri, R. A. Hamid, A. R. Abdulnabi, O. S. Albahri, A. H. Alamoodi, M. Deveci, et al., Fuzzy decision-making framework for explainable golden multi-machine learning models for real-time adversarial attack detection in vehicular ad-hoc networks, *Inform. Fusion*, **105** (2024), 102208. <https://doi.org/10.1016/j.inffus.2023.102208>
30. C. Joshi, J. Kumar, G. Kumawat, Detection of unseen malware threats using generative adversarial networks and deep learning models, *Sci. Rep.*, **15** (2025), 34804. <https://doi.org/10.1038/s41598-025-18811-3>



AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)