SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I6P118

Volume 13 Issue 6, 218-247, June 2026

Original Article

ApR-FM2TEDL: Enhanced Tri-Ensemble Optimized Deep Learning Framework for Attack Detection and Mitigation in Cloud Networks

Yogesh B. Sanap[1], Pushpalata G. Aher[2]

[1,2]*Computer Science and Engineering, Sandip University, Nashik, Maharashtra, India.*

[1]*Corresponding Author : sanap.yogesh@gmail.com*

Received: 20 March 2026 Revised: 19 April 2026 Accepted: 18 May 2026 Published: 27 June 2026

Abstract *- The usage of web services has increased the volume of data across cloud computing systems with a larger range of users. Cloud networks provide convenient access to users with storage facilities for evolving circumstances. However, the distributed nature of cloud networks, as well as the increasing data volume, leads to various attacks that affect the integrity of the stored information. Several methods have been presented for the detection of attack networks; however, they are vulnerable to limitations, including lower accuracy, longer training time, overfitting issues, and higher complexities. Consequently, the research overcomes these limitations with the Proposed Apis Random movement optimized Fuzzy Min-Max Enabled Tri-Ensemble Deep Learning (ApR-FM2TEDL) framework that exhibits accurate detection. The incorporation of the Apis Random Movement Optimized Synthetic Minority (ApRO-SyM) sampling method produced synthetic samples over the minority class that prevent the overfitting issues. Specifically, the Apis Random movement Optimization (ApRMO) Algorithm carries out hyperparameter tuning and improves the data generation by choosing better neighbor sets. The effectiveness of the model is computed using training percentage, which shows improvements in the sensitivity of 96.20%, accuracy of 97.74%, and specificity of 98.59% for the BOT-IoT dataset. Further, the proposed ApR-FM2TEDL framework achieves a superior accuracy of 97.13%, sensitivity of 98.56%, and specificity of 96.72% with 80% of training using the CIC-DDoS2019 dataset, outperforming the other baseline models.*

Keywords *- Attack Detection, Deep Learning, Synthetic Sampling, Cloud Computing, Traffic Flow.*

1. Introduction

The rapid increase of cloud computing has provided the greatest standard of information technology over the past few years, with a wide range of network users. The improvements in existing network interfaces include distributed and parallel computing, where cloud networks are proven to be efficient [1]. The computing network provides consumers with increased resources and storage facilities with a convenient infrastructure for dynamic environments [2]. The cloud computing resources enable users or organizations to reduce the infrastructure cost by choosing different online resources that are in the form of services [3]. With the convenience in management or the interaction between the resource providers, the networks provide configurable computing with a shared pool of connections [4]. Cloud computing and distributed systems are two different devices in which the distributed system distributes the data in several places, and it can be retrieved from anywhere in the world. Accordingly, in cloud computing, all information is stored on the Internet and can be accessed remotely from anywhere [5, 6]. Cloud systems mainly face attacks in the network layer, namely spoofing or scanning attacks, Denial of Service (DoS), and Distributed DoS attacks [7 Intrusion tools have become more challenging in the existing Cloud systems as they require large volumes of network traffic data, dynamic and complex behaviors, and new types of attacks [8]. Generally, the traditional intrusion detection systems use signature-based techniques as one of their defensive strategies to detect known attacks by comparing traffic patterns to established attack signatures [9, 10]. However, firewalls cannot detect intrusions since they are designed only for capturing network packets within specific boundaries of the computing network [11].

Furthermore, some of the conventional traffic-based attacks are highly advanced for the firewalls to detect, hence the research has shifted towards the development of Network Intrusion Detection Systems (NIDS) [12]. The systems can safeguard the network from intrusions, serve as an alert mechanism, and act as the cybersecurity strength of the entire association [13]. The IDS is a cornerstone in the detection as well as mitigation of security breaches that penetrate the system. Artificial Intelligence-based detection systems exhibit

highly efficient performance by incorporating the Machine Learning (ML) approaches, including decision trees (DT) [14]. Recently, the Machine Learning (ML) approach employs a variety of algorithms to address the security challenge and better manage data. Most of the datasets are private and cannot be released owing to privacy concerns, or they may be missing important statistical properties [15]. In addition, the larger feature space led to overfitting issues, as well as preventing the models from exploiting all the feature vectors. The development of effective systems is challenging; however, it is crucial for improving the security measures in the ever-growing era of digital attacks. In complex environments, Deep Learning (DL) technology is employed to promote the possibility of processing larger volumes of unstructured data with accurate recognition [16]. The methods identify the non-linear relationships as well as the sequential patterns of the intrusion packets leading to potential detection [17]. The DL methods are highly utilized in IDS experiments due to their ability to validate the network traffic in real time and mitigate the attacks [8].

Despite the fact that DL methods offer better performance, these methods are found with high false alarm rates and face difficulty in detecting evolving attack threats. However, the efficiency of these methods is impacted regularly because of the ineffective nature of utilizing a wide range of resources [18]. Besides, the dynamic characteristics of cloud systems, including the resource allocation, rapid scaling, and workload changes, often complicate the task of maintaining the resilience and adaptation over time, limiting their performance [8]. The real-time IDS methods necessitate extensive computational resources with the increase in the data volume, which results in a reduction in efficiency [19]. The outcomes of the DL models are often questioned due to the potential vulnerabilities in unexpected conditions [5, 20]. The ideal solution for improving efficiency is maintaining the effectiveness in analyzing the real-time data traffic, as well as working on limited resources [21]. Moreover, the utilization of such systems can increase the sustainability regarding the computational costs as well as the agility in rapidly evolving cyber environments [22]. One of the most significant issues is the inherent complexity and variety of cloud data, which includes distributed sources such as system logs, network traffic, and user behavior. Additionally, the dynamic nature of cloud systems, which includes resource allocation, rapid scaling, and workload changes, complicates the task of maintaining model resilience and adaptation over time. Addressing these challenges requires interdisciplinary efforts to develop scalable, resource-efficient, and privacy-aware methods.

Therefore, this research proposes an ApR-FM2TEDL framework that successfully distinguishes the type of attacks in the cloud network data. The method subdues the limitations regarding overfitting, lower accuracy, complexity, and misdetections that affect the performance of the conventional approaches. The ApRMO algorithm improves the detection with the hyperparameter tuning as well as the minority sample generation. Moreover, the DL layers interpret the dynamic nature of attack-related features, and the identified attack packets are mitigated using the mitigation module. The main contributions of this work are discussed as follows.

Apis Random Movement Optimization: The ApRMO algorithm is designed for the weights tuning in the ApR-FM2TEDL model and to enhance the selection of quality k-neighbors for generating better synthetic data inspired by the random movement and foraging strategies of the bee colonies for finding solutions in the local optima regions and improving the convergence.

Apis Random movement optimized Synthetic Minority Sampling: The development of the ApRO-SyM sampling method reduced the class imbalances with the generation of synthetic data. The ApRMO algorithm is incorporated to choose effective neighbors for improving the standard of samples, which in turn prevents the ApR-FM2TEDL system from exhibiting biased detections.

Apis Random movement optimized fuzzy min-max enabled Tri-ensemble Deep Learning method: The ApR-FM2TEDL method encompasses three layers of parallelly working DL networks that promote the learning of composite features. Furthermore, the model produces accurate detection results for the attacks in cloud data with hyperparameter tuning using the ApRMO algorithm.

The manuscript is organized as follows: Section 2 provides the background of the related approaches implemented for cloud attack detection, Section 3 explains the system model of the attack detection system, Section 4 discusses the methodology of the ApR-FM2TEDL approach, the experimental outcomes of ApR-FM2TEDL are analyzed in Section 5, and Section 6 includes the conclusion and future direction of the research work.

2. Literature Review

In this section, the overview of existing methods is discussed with the employed methods, their benefits as well as their limitations. Numerous studies were conducted for detecting attacks in cloud networks. This research focuses on the DL techniques, including transformer-based and classical ML-based methods, which are explained in detail with their challenges.

Mohamed Ouhssini et al [9] suggested Deep Defend, which was employed to identify DDoS incursions in cloud systems. The transformer-based method predicted potential attacks by analyzing the traffic in the network. The accuracy of the approach was higher for entropy forecasting, which was achieved through the training based on time-series traffic data. The complexities with resources, which were minimized,

however, exhibited lower adaptability with unpredictable conditions. Syed Mohamed Thameem Nizamudeen, [8] presented an intelligent IDS framework that identified the attacks from multi-cloud servers with the selection of significant attributes. The transformation process of the data was performed using grading normalization, where the stability was given higher priority. The method distinguished the abnormalities from non-malicious data as well as identified the occurrence of adversarial attacks in individual layers. Nevertheless, the heterogeneity of the cloud systems made it difficult to maintain stability between different traffic variables.

Wa'ad H. Aljuaid and Sultan S. Alshamrani [2] introduced an advanced Convolutional Neural Network (CNN) based system that detects cyber-attacks from a wide range of cloud data. The crucial features were acquired using the feature selection technique and performed sampling that minimized the imbalances. The categorical values were simplified with the application of a label encoder, whereas the false positives were higher in dynamic computing systems. Zhenyue Long, et al [4] modeled a transformer-based approach that exhibited higher adaptability with the evolving threat vectors. The inclusion of an attention mechanism facilitated the observation of the relationships between the intrusion features. The encoder-decoder structure interpreted the varying length of sequences; however, the method faced difficulties in analyzing the complex sequences over distributed environments.

Muhammad Sajid et al [19] modeled a DL detection technique that classified the various network threats affecting the cloud environments. The method involved extracting the spatial as well as temporal characteristics from a wider range of data. The detection speed of the DL model was higher with greater interpretability for composite attack sequences. However, the method faced higher challenges due to data imbalances in specific classes. Abdel-Rahman Al-Ghuwairi et al [23] suggested a selection-based approach that employed Analysis through time-series data with intrusion detection. The method performed continuous observation for tracking the changes in security specifications and provided a prior warning of the manifestation of potential threats. The collaborative selection of features improved the accuracy, whereas the lack of dissimilarity measures increased the complexity.

Nishika Gulia et al [17] explained a Group Artificial Bee Colony (G-ABC) that manifested effective performance in observing the intrusions in cloud services. The method handled the non-numerical features efficiently and performed normalization with maximum to minimum values. The traffic in the network was captured using a time sequence-based window that delivered better results in detecting the sources of anomalies. However, the G-ABC approach was limited in detecting specific cloud attacks. Mehmood, M. et al., [15] introduced a machine learning approach to detect and classify attacks in the cloud network. The utilization of an ensemble learning approach improves the performance of the model. The model demonstrates higher accuracy with better detection performance. High computational cost and limited datasets were the major drawbacks faced by the model.

2.1. Challenges

- Some of the related research works with drawbacks are listed below,
- The 2D-CNN-based model lacked training data and faced vulnerabilities in the composite network topology. In addition, the evaluation of the method was not performed using real-time attack data [8].
- The adaptability of the Deepdefend approach was lower under unpredictable breaches and failed to capture the intrinsic relationships between the network packets [9].
- The lack of optimization methods affected the efficiency against the adverse network traffic. Furthermore, the evaluation of sequential data was not performed, leading to potential impacts in altering circumstances [2].
- The DL method exhibited higher complexity with a longer training period and higher false alarm rates [19].

2.2. Problem Statement

Cloud computing provides defined computing with virtually shared resources and workstations, facilitating rapid services as well as larger storage. Typically, the distributed and heterogeneous nature of the network opens the way for intrusions that disrupt cloud services based on trust and reliability. Some of the traditional methods exploited for intrusion detection are subjected to profound complexities regarding limited generalizability, complexity [8], as well as overfitting issues [15, 23]. In [19], the imbalances in the data distribution led to a biased outcome as well as higher false positives. In addition, the lack of preprocessing as well as the inefficiency in the interpretation of complex features resulted in misdetections [9].

The research overcame the limitations with the development of the ApR-FM2TEDL framework, which detected the various networks from normal traffic. In addition, the ApRO-SyM sampling reduced the data imbalances with the generation of optimal samples. The IoT database W containing the attack and normal traffic information is represented as,

$$W = \left(W_{i,j}, Lb_i\right); \begin{cases} 1 \le i \le Y \\ 1 \le j \le X \end{cases} \quad (1)$$

Here $Lb_i$is the label respective to the i^{th}row; YX denotes the minimum and maximum number of rows in the dataset, the information about the packets transferred over the internet at a specific time is denoted by i^{th}rows, and j^{th}column

represents the data attribute of the particular i^{th}row. The labels in the datasetLb can be signified as,

$$Lb_i \in (0,1), 1 \leq i \leq Y \quad (2)$$

The ApR-FM2TEDL method effectively identifies the presence or absence of attacks in the data, which is denoted by

$$Lb_i = \begin{Bmatrix} 0, Normal \\ 1, Attack \end{Bmatrix} \quad (3)$$

The research utilizes Binary cross-entropy loss for improving the accuracy of ApR-FM2TEDL in accurately predicting the attack and normal data, which is denoted as

$$Lo_{BCE} = -\frac{1}{Y}\sum_{i=1}^{Y} At_i \, log(Pd_i) + (1 - At_i) \, log(1 - Pd_i) \quad (4)$$

Here, At_iindicates the actual label of the i^{th}sample, and $Pd_i$is the predicted label of the i^{th}sample. The ApR-FM2TEDL provides higher efficiency in identifying the various attacks from the cloud data.

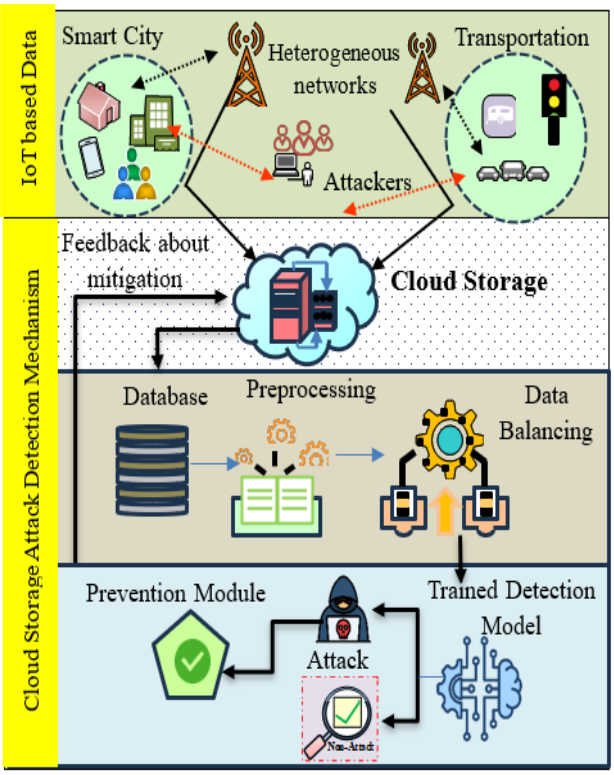


Fig. 1 System model of the attack detection and mitigation in cloud networks

3. System Model for Threat Detection in Cloud Networks

Across distributed cloud infrastructures, the identification of threats or intrusions is a significant research hotspot that provides the security of resources by potentially mitigating the attack packets. With the identification of suspicious behavior, the attack detection models are highly considered essential among suppliers of Infrastructure As A Service (IaaS) that allows resources, including virtual machines on demand.

The packets are either benign or malicious and are collected from the network traffic in the cloud and fed into the method for detection. The systems interpret the authenticity of the data packets, and the mitigation module eliminates the packets identified as threats and updates the information. The updation of threat information to the model is essential for blocking the same packets in the future from exploiting the network resources.

Moreover, the utilization of data balancing, as well as preprocessing, promotes the detection performance by enhancing the data standard. The development of a versatile and efficient detection model is essential to identify the potential threats accurately in changing environments. Figure 1 represents the system model of the attack detection module for cloud computing environments.

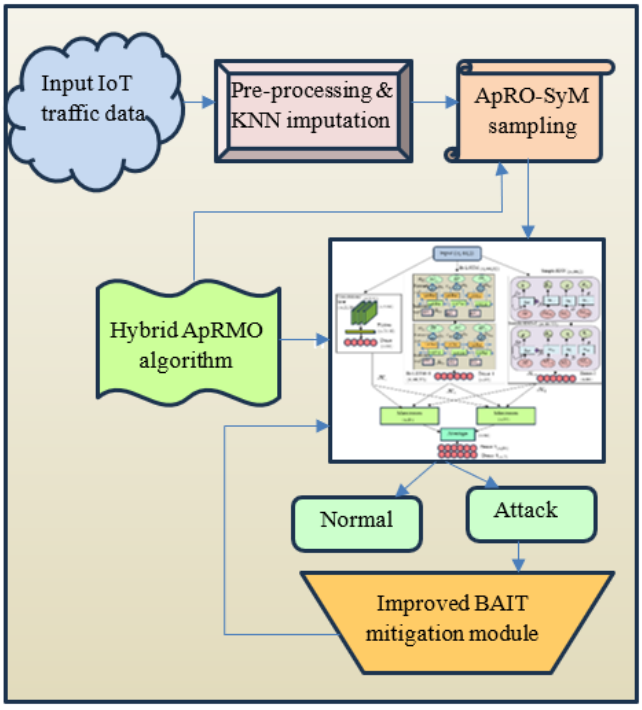


Fig. 2 Schematic Representation of the ApR-FM2TEDL framework for Attack detection

4. Proposed Apr-Fm2tedl Framework for Attack Detection and Mitigation in Cloud Networks

The main objective of this research is to mitigate and detect the security concerns in cloud environments using the proposed ApR-FM2TEDL framework. The schematic illustration of the proposed work is visualized in Figure 2. Initially, the research sources network data from Bot-IoT and the CIC-DDoS2019 Dataset, which undergoes preprocessing

where the missing values in the data are imputed using KNN imputation. After the imputation, the preprocessed data is fed into the ApRO-SyM sampling module that balances the imbalanced distribution of classes. The application of the ApRMO algorithm enhances the generation of better samples by assisting in the selection of quality neighbors. Followed by sampling, the data is given to the ApR-FM2TEDL for the effective detection of attacks in the data, where the hyperparameters are tuned using the ApRMO algorithm. If the model detects the presence of attacks, then the improved BAIT mitigation module mitigates the attacks from the network and updates the information to ApR-FM2TEDL for identifying similar attacks in the future.

4.1. Input Data for Attack Detection

To detect attacks, the input data in this research are sourced from CIC-DDoS2019 and the Bot-IoT Datasets that contain the system traffic acquired from various heterogeneous environments. The input data to be processed is mathematically expressed as,

$$W = \{W_1, W_2, \dots W_i \dots W_Y\} \quad (5)$$

Here, W denotes the dataset, $\{W_1, W_2, \dots W_i \dots W_Y\}$ represents the total data, and $W_i$is the input data. The input acquired from CIC-DDoS2019 is in the dimension$(n, 431371{,}79)$, and the BoT-IoT Dataset is in the dimension $(n, 175341{,}44)$.

4.2. Preprocessing using K-Nearest Neighbor Imputation

The data gathered from the IoT networks are subject to inconsistencies, missing values, and noise artifacts, which need to be minimized for generating meaningful detections. The data is preprocessed in this research using the K-Nearest Neighbor Imputation (KNN) technique, which imputes the missing data by replacing values based on the neighborhood values [24].

The method selects the distant vector and computes the KNN values closer to the estimated vectors to find the better values. The average nearest point between the nearest set and the estimated vector is evaluated to predict the missing value [25]. Then the missing instances are filled with the proximity values corresponding to the neighbors, which minimizes the complexities in detection. The preprocessed data is represented as W_i^*, which is in the dimension of $(n, 431371{,}79)$.

4.3. Data Balancing using APIs, Random Movement Optimized Synthetic Minority Sampling Technique

The imbalances in the data affect the detection outcomes of the model and produce biased results due to the overfitting issues. The standard method, namely the Synthetic Minority Oversampling Technique (SMOTE), is utilized in various related works to deal with the imbalances. The method minimized the complexities to a certain extent, while the random selection of neighbors, interferences, and the overlapping issues results in samples with lower quality [26]. To overcome the limitations and to generate effective samples, the research proposed an Apis Random Movement Optimized Synthetic Minority (ApRO-SyM) sampling technique. The appropriate selection of neighbors is the major factor for effective generation since the selection of lower K-neighbors leads to underfitting and the higher values result in overfitting of the data [27]. The application of the developed ApRMO algorithm improves the synthetic sample generation by assisting in the selection of standard K-neighbors. The method adopts the search characteristics that provide equal priority during exploitation and exploration for selecting better neighborhood values.

The ApRO-SyM assists in selecting the number of synthetic data points to be generated in each class based on the sample rate parameter that manages the oversampling to be performed. The sample rate parameter ϛcontrols the number of generated synthetic samples, which is optimized by the ApRMO algorithm. The generation of high-quality synthetic data prevents overfitting issues and boosts the performance of ApR-FM2TEDL in detecting attacks. The samples generated by the ApRO-SyM sampling method are represented as Sa^{new}, which are in the dimension of $(n, 667080, 79)$ and are given into the proposed ApR-FM2TEDL framework for attack detection.

4.4. Proposed APIs Random MOVEMENT OPTIMIZED FUZZY Min-Max Enabled Tri-Ensembled Deep Learning Framework for Attack Detection

In the proposed approach, the Fuzzy min-max enabled Tri-ensembled Deep Learning framework is utilized for achieving efficient attack detection. Utilizing the Tri-ensemble Deep learning method, combining Deep CNN, Bi-LSTM, and RNN, enhances the feature representation to explore the hidden attack patterns, and the element-wise min-max operation facilitates obtaining the maximum and minimum data features, which are acquired separately. Moreover, the incorporation of the advanced mechanisms improves the architecture to handle the uncertainty in the input samples, enhances the training process, and provides improved attack detection.

The architecture of the ApR-FM2TEDL model developed for threat detection in cloud networks is shown in Figure 3. Several research works have been deployed for cloud threat detection; however, they are vulnerable to limitations including higher complexities, overfitting, and limited generalizability [17, 19]. The ApR-EM2TEDL framework overcomes the limitations of prevailing methods by exhibiting effective attack detection.

The ApR-EM2TEDL comprises three effective deep learning models, namely Deep CNN, Bi-Directional Long Short-Term Memory (Bi-LSTM), and Recurrent Neural

Network (RNN), where the outputs from the three models are fed into the fuzzy minimum-maximum averaging technique. The ApR-EM2TEDL is efficient in detecting attacks from normal packets and works as a parallel processing unit.

The input Sa^{new} is fed into the three parallel working DL models simultaneously to streamline the detection process. Initially, in the Deep CNN layer, the input Sa^{new} from the ApRO-SyM sampling method passes through the convolutional layers, which contain several kernels that extract the essential features [28]. Some of the features from the generated data are captured using the filters, and an activation map χ_{ma} is acquired, which is expressed as,

$$\chi_{ma} = relu(Conv(Sa^{new})fe_{cn}, fa_{cn}) \quad (6)$$

Where fe_{cn}, fa_{cn} represents bias and weights of the convolutional filter, $Conv$ is the convolution function, and $relu$ represents activation that introduces non-linearity between the layers. The χ_{ma} multidimensional data that is flattened based on a single dimension using a flattened layer and dense layer $(n, 64)$ is a fully connected network that captures the details attained from previous layers and generates an outcome JK_1 in the dimension $(n, 64)$. Meanwhile, the input Sa^{new} is given into the Bi-LSTM layer $(n, 44{,}32)$, which passes the data in forward (rd) and backward (wd) directions for analyzing the crucial relationships [29]. The memory cells store the information and maintain the data flow, where the input gate EN_t obtains the input to update the cell state, which is denoted by

$$EN_t = \sigma\left(fe_{i1}Sa^{new} + fe_{i2}dn_{t-1} + fa_{I_t}\right) \quad (7)$$

Here, $fe_{i1}fa_{I_t}$ implies weights and bias in the input gate layer, σ which denotes the sigmoid function, and dn_{t-1} is the hidden layer of the past iteration. The unwanted details from the data are removed by the forget gate output generated by the output gate with the updation of information.

$$EP_t = \sigma\left(fe_{u1}Sa^{new} + fe_{u2}dn_{t-1} + fa_{U_t}\right) \quad (8)$$

Where the weights and bias are represented by fe_{u1} and fa_{U_t}. Accordingly, the result from the Bi-LSTM layer contains information obtained from both directions, denoted as

$$JK_1^{Bi} = \left[\overrightarrow{EP_t} * \overleftarrow{EP_t}\right] \quad (9)$$

Here $\overleftarrow{EP_t}$ are the outputs of the forget gate obtained from forward and backward directions. Moreover, the resultant JK_1^{Bi} is forwarded to the Bi-LSTM-1 layer, where a similar process is performed, and output $JK_2^{Bi}(n, 44{,}32)$ is attained. The dense-1 layer gathers the input and generates it $JK_2(n, 64)$.

$$JK_2 = \left[JK_1^{Bi} + JK_2^{Bi}\right] \quad (10)$$

In the meantime, the input Sa^{new} is also passed through the RNN layer, which analyzes the sequential components of the data to identify the semantic relationships. The input layer perceives the information where the data is transferred from one iteration to another and predicts the output corresponding to the past details. The non-linear relationships of the threat data are learned with the addition of weights and biases with the tanh activation to predict the temporal dynamics [30]. The hidden layers upgrade the information at each iteration in the output gate B , and the information from the present time step is declared as output JK_{rnn} that can be mathematically denoted as,

$$B = tanh(fe_{rn} \times he_t + fa_{rn}) \quad (11)$$

where, fe_{rn}, fa_{rn} indicates the weights and bias of the output layer, and the output JK_{rnn} from the simple RNN-1 layer passes through dense-2 and generates $JK_3(n, 64)$. The outputs JK_1, JK_2, JK_3 pass through both maximum and minimum layers, $(n, 64)$ in which element-wise min-max operation is performed, where the maximum and minimum data features are acquired separately. The extracted features undergo a weighted average of the features that are computed and passed to dense-3 and dense-4 layers to mitigate the attacks in the data. The research included the ApRMO algorithm utilized in the fourth layer that tunes the weights of the model, leading to enhanced threat detection. The data packets identified as threats are sent to the mitigation module to eliminate the malicious contents from the network. Figure 3: Architecture of the ApR-FM2TEDL model.

4.5. Apis Random Movement Optimization

In the proposed research, the ApRMO algorithm is employed for optimally fine-tuning parameters of the ApR-FM2TEDL model and to improve the generation of better samples in the ApRO-SyM Sampling method. The ApRMO is simulated based on the behavioral characteristics of coots and bee colonies to reach the optimal solution. The characteristics include random motion, position updation, and foraging, and the utilization of these attributes improves the global search capability [31]. Although the existing optimization techniques, including Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Grey Wolf Optimization (GWO), are widely used, their performance is inadequate because of multifaceted and voluminous data. Meanwhile, the Coot Optimization Algorithm (COA) suffers from shortcomings such as slow convergence and getting trapped in local optima. Further, the ABC algorithm exhibits better performance, but the inherent limitation of insufficient balance between exploration and exploitation restricts the applicability of the algorithm. In contrast, the ApRMO algorithm addresses the issues regarding the complexities,

convergence, as well as suboptimal convergence problems, and provides the global optimal solution. More specifically, the proposed ApRMO algorithm offers effective global searching ability and ultimately reaches the optimal solution depending on previous experience. In addition, the ApRMO algorithm achieves global convergence for hyperparameter tuning of the ApR-FM2TEDL model and improves the ApRO-SyM Sampling, resulting in an increase in the attack detection performance.

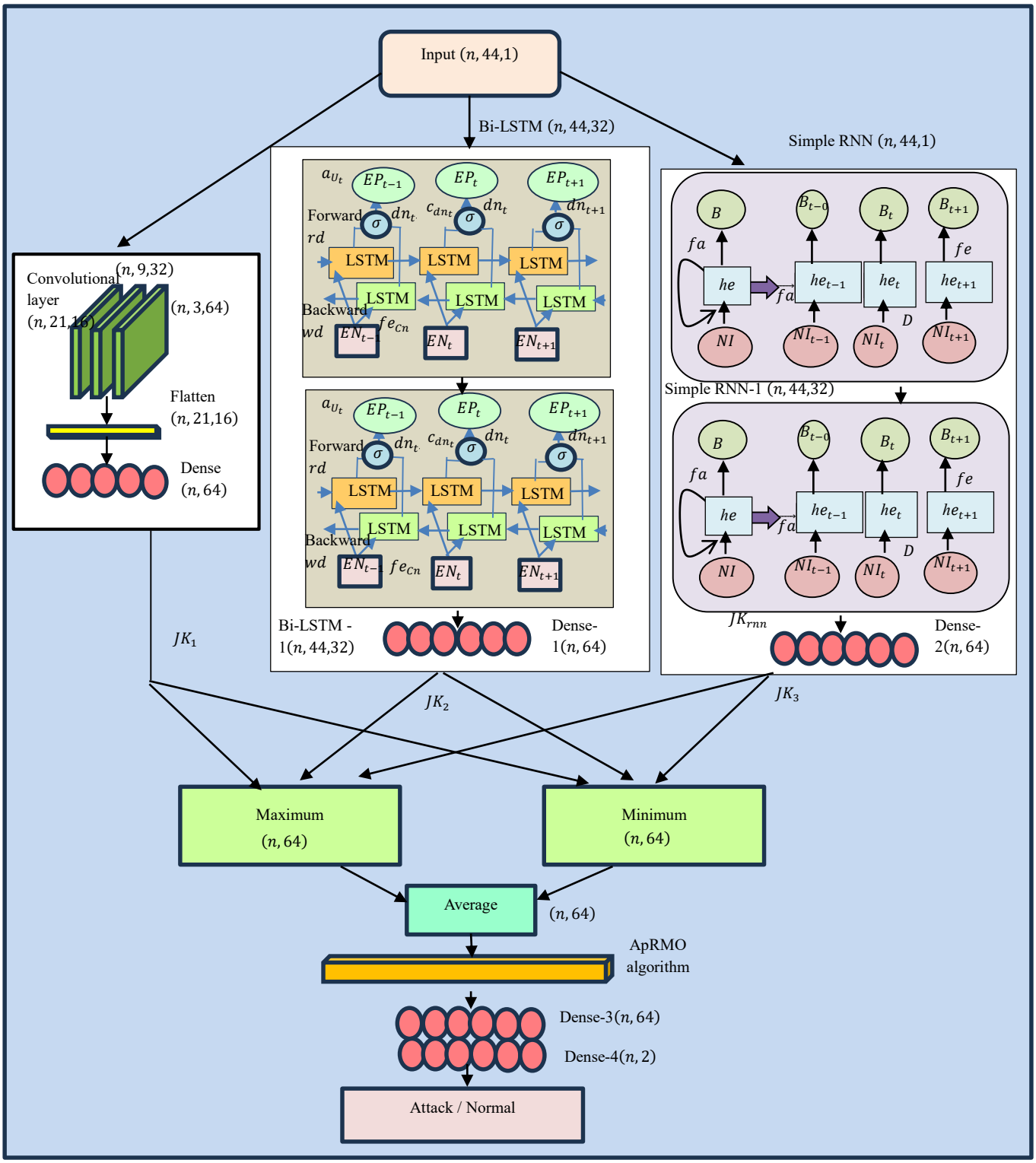


Fig. 3 Architecture of the ApR-FM2TEDL model

Inspiration: The ApRMO algorithm draws its characteristics from the swarm behavior of coots and bee colonies, which includes the position updation, hunting, and chain motion. The exploration behavior seen in bees promotes an accurate capability for global search of solutions. The random motion and the tuning of parameters facilitate deep exploration in the search space for attaining optimal solutions [32]. The ApRMO provides equal priority with search space during exploitation and exploration, which prevents the acquisition of solutions from the local optima region. The developed ApRMO algorithm enhances the sampling performance with the selection of optimal neighbors.

The bees search for food sources as colonies in a wider range to collect the nectar, which is adopted in ApRMO to find better solutions where the food represents the possible solutions and the amount of nectar denotes the quality of corresponding solutions. The exploitation continues until the sources are depleted, and the coots' behaviors involve the chain motion around the leaders. They exhibit a unique strategy with position updates, which assists the ApRMO in updating the positions during the scarcity of optimal values or the presence of attacking solutions.

Solution Initialization: The solutions are initialized for the performing search in the search space, where the population represents the solutions with the associated decision variables. The initialized solutions can be mathematically denoted as,

$$R_u: r_{u,v} = lp_v + rand(Up_v - lp_v) \tag{12}$$

Where $u = 1,2 \ldots . P$the total solutions are denoted by P, and the search space dimension is Q, the initial position vector of $u^{th}$the solution is R_u, $rand$denotes the random number in the interval of $(0,1)$, $lp_v$and $Up_v$are the lower and upper bounds of v^{th} the search space. The population solutions can be mathematically signified as,

$$R = \begin{bmatrix} R_1 \\ : \\ R_u \\ : \\ R_P \end{bmatrix}_{P\times 1} = \begin{bmatrix} r_{1,1} \ldots . r_{1,v} \ldots . r_{1,Q} \\ : \\ r_{u,1} \ldots . r_{u,v} \ldots . r_{u,Q} \\ : \\ r_{P,1} \ldots . r_{P,v} \ldots . r_{P,Q} \end{bmatrix}_{P\times Q} \tag{13}$$

Fitness Function: The effectiveness of the solutions is computed through the objective function, where the higher quality of solutions has a larger fitness value. The fitness function of the proposed ApR-FM2TEDL is computed using accuracy, and that can be represented as,

$$Fn\left(R_{u,v}^{g}\right) = max\left(Accuracy\left(R_{u,v}^{g}\right)\right) \tag{14}$$

The fitness of the APRMO-SyM model is measured based on the F1 score that can be expressed as,

$$Fn\left(R_{u,v}^{g}\right) = max\left(F1score\left(R_{u,v}^{g}\right)\right) \tag{15}$$

Case (i) Global search phase: if $Nd \geq 0.5$

The global search process occurs when the value $Nd$is greater than or equal to 0.5. Here, Ndrepresents the probability of the random value being present in a global search that belongs to the range between $(0,1)$. The updated position based on the chain movement of the coots that is represented by,

$$R_u^{(g+1)} = 0.5\left(R_{u-1}^{g} + R_u^{g}\right) \tag{16}$$

Where represents the current position vector of the u^{th} solution, and $R_{u-1}^{g}$is the position of the previous solution. The conventional methods focus on initializing the best solution; however, there exists a chance that the solutions may be selected as best from the local optima, which will not be effective. Therefore, this research selects random values by giving equal priority between the local and global search for finding better values depending on the foraging behavior of artificial bee colonies. Here, the updated position is denoted as,

$$R_u^{(g+1)} = R_u^{g} + \varphi_u\left(R_u^{g} + R_{rand}^{g}\right) \tag{17}$$

Where $R_{rand}^{g}$is the position of the random solution, and $\varphi_u$is the randomly selected number, which ranges between $(-1,1)$. The position update, depending on the random solution, is represented as,

$$R_u^{g} = \frac{1}{(1+\varphi_u)}\left[R_u^{g+1} + \varphi_u \times R_{rand}^{g}\right] \tag{18}$$

Further, the solution's current position is updated in the chain motion-based position update equation to improve the probability of acquiring optimal solutions. The position updation is dependent on the randomly selected number and scaling parameters, which can be mathematically signified as,

$$R_u^{g+1} = \delta\left[\gamma \times R_{u-1}^{g} + \varphi_u R_{rand}^{g}\right] \tag{19}$$

Where, $\delta$and$\gamma$are scaling parameters which are given by $\delta = \left[\frac{1+\varphi_u}{0.5+\varphi_u}\right]and\gamma = 0.5$. The utilization of tuning parameters in the above equation provides better exploration in search regions, which minimizes the time complexity and leads to faster convergence.

Case (ii) Local Search: if$Nd < 0.5$

The solution undergoes local search based on the probability that the random value presented in the global search$Nd$is less than 0.5. The local search is followed based on the knowledge acquired by the neighbors, where the information from better neighbors can be effective, while the non-optimal neighbors reduce the probability of better

extraction. Therefore, optimal leaders are selected based on their effectiveness as well as alertness with search space conditions. Depending on the movement of the selected leaders, the solutions follow two movements, namely guided and unguided motions.

Subcase (i) Unguided Phase: if $HG \leq 0.5$

The unguided phase is employed when the guided parameter HG is less than or equal to 0.5, and the value HG ranges between $(0,1)$. The updated position depends on the best position and the constant parameter that can be expressed by,

$$R_u^{g+1} = R_u^g + Ad \times abs\left(R_{best}^g - R_u^g\right) \tag{20}$$

Where Ad is the control parameter that is essential for controlling the tradeoffs between the search space that can be obtained based on the iterations and current values? The parameter Ad can be mathematically signified as,

$$Ad = Ct \times e^{-\left(\frac{2\times g}{g_{max}}()\right)} \tag{21}$$

Where Ct represents the constant parameter with the fixed value of 1.5, abs denotes the absolute operation, g and g_{max} are the current and maximum iterations.

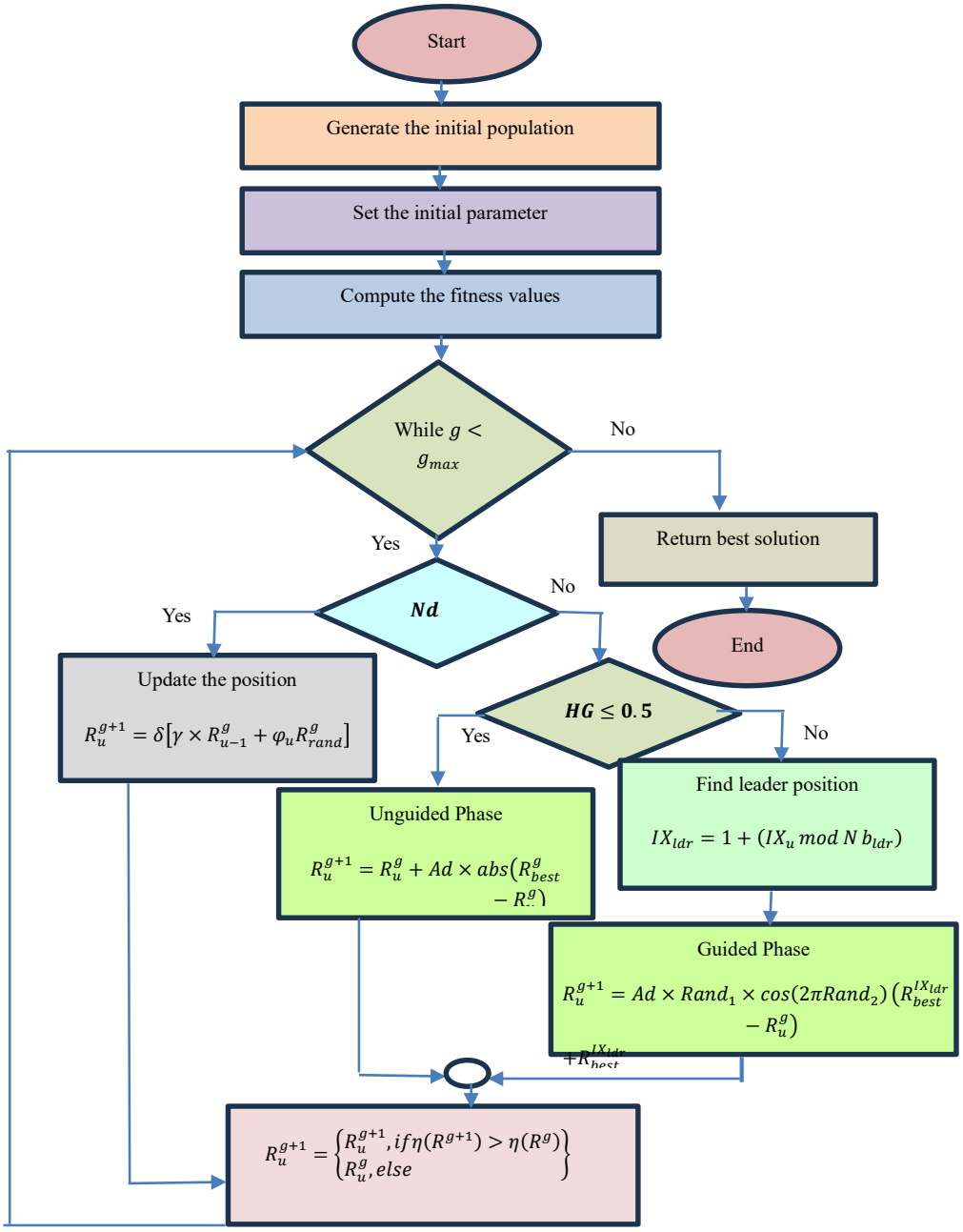


Fig. 4 Flowchart of the ApRMO algorithm

Subcase (ii) Guided Phase: if$HG > 0.5$

The solutions undergo searching with the guidance of the leader solutions when the value of the guidance parameterHG is greater than 0.5. The position of the leader is evaluated depending on the index of the current solution, which can be denoted by

$$IX_{ldr} = 1 + (IX_u \, mod \, N\, b_{ldr}) \quad (22)$$

Where IX_{ldr}signifies the index value of the leading solution, Nb_{ldr}denotes the number of leading solutions, which is equal to 1, IX_udescribes the index of the current solution, and modrepresents the modulus operation. Typically, the updated position is denoted as,

$$R_u^{g+1} = Ad \times Rand_1 \times cos(2\pi Rand_2)\left(R_{best}^{IX_{ldr}} - R_u^g\right) + R_{best}^{IX_{ldr}} \quad (23)$$

Here $R_{best}^{IX_{ldr}}$is the current best position of the leading solution: the random values in the ranges between $(0,1)$and $(-1,1)$, cosdenotes the cosine, and the value of $\pi$is 3.14. The cosine function maintains the movement of the algorithm in a controlled environment with adjustments to the search positions. In addition, the estimation of a better position provides a precise path towards the optimal solution. The ApRMO algorithm finds the weakest weights and updates them using position updates to generate the highest fitness value of each solution.

Termination Condition: The working of the ApRMO algorithm ends once the value $g > g_{max}$is obtained and returns the global best solution. Figure 4 illustrates the flow diagram of the ApRMO algorithm.

In order to compare the strength of the ApRMO algorithm with existing algorithms, the pseudocode of the COOT optimization, ABC algorithm, and ApRMO algorithm showing the implementation steps to reach the best solution are detailed in the Algorithms 1,2, and 3 as follows,

Algorithm 1. Pseudocode of COOT Optimization

S.NO	COOT Optimization
1	Initialize the parameters $ML, Mcoot, and M$
2	**Output:**
3	Initialization: $R_{u,v}^g = rand(1,h) * (U_b - L_b) + L_b$
4	Here $Mcoot = M - ML$
	Select MLa random number of solutions.
5	Evaluate the fitness of each solution.
6	Find the best solution. Z_{best}
9	If $(t < t_{max}())$
10	{
11	Compute the random function$A$,B, and generate the random values.r_1, r_2, r_3, r_4 $A = 1 - t \times \left(\frac{1}{t_{max}}() \left(\frac{1}{t_{max}}()\right)\right)$
13	for each coot R_i, do,
14	Find the index number of the elected leader K_{ind}
15	if $(rand > 0.5)\{$
16	Update the position using.$R_u^{(g+1)} = R_K lead + 2 \times r_2 \times cos(2\pi\ell) \times \left(R_K lead - R_i(t)\right)$
15	if $(rand < 0.5)\{$
17	Update the position using. $R_i(t+1) = \frac{(R_{i-1}(t) + R_i(t))}{2}$
18	else
19	$R_i(t+1) = R_i(t) + A \times r_1 \times \left(R_{rand}(t) - R_i(t)\right)$
20	end if
21	Evaluate the fitness solution.
22	if $fit(R_i) < fit(R_K lead)$
23	swap$(R_i, R_K lead)$
24	end if
25	end for
26	for each leader $R_i lead$do
27	Update the position of the leader. $R_i lead(t+1) = \begin{cases} B \times r_3 \times cos(2\pi\ell) \times \left(R_{best}^{IX_{ldr}} - R_i lead(t)\right) + R_{best}^{IX_{ldr}} r_4 < 0.5 \\ B \times r_3 \times cos(2\pi\ell) \times \left(R_{best}^{IX_{ldr}} - R_i lead(t)\right) - R_{best}^{IX_{ldr}} r_4 \geq 0.5 \end{cases}$

28	if$fit(R_i lead < R_{best}^{IX_{ldr}})$
29	$(R_i lead, R_{best}^{IX_{ldr}})$
30	end
31	end for
32	$t = t + 1$
33	Return $R_{best}^{IX_{ldr}}$

Algorithm 2. Pseudocode of ABC Optimization

S.NO	Artificial Bee Colony Optimization
1	Initialize the parameters
2	Initialize the random solution: $R_{i,j} = L_{i,j} + rand(0,1) \times (U_{i,j} - L_{i,j})$
4	Evaluate the movement of onlooker bees.
5	$R_{i,jr}^{new} = R_{i,j} + \delta_{i,j} \times (R_{i,j} - R_{k,j})$
6	Where $R_{i,jr}^{new}$ is the new position
7	Probabilistic selection of onlooker bees
8	$P_i = \frac{F_i}{\sum_{i=1}^{N} F_i}$
9	Update the best solution with the new solution:
10	if $fit(R_{i,jr}^{new}) < fit(R_{i,j})$, then $R_{i,j} = R_{i,jr}^{new}$
11	Memorize the best solution. $R_{best}^{IX_{ldr}}$
12	$t = t + 1$
13	Return Z_{best}

Algorithm 3. Pseudocode of Apis Random Movement Optimization

S.NO	Apis Random Movement Optimization
1	**Input:** Initialize the parameters$R_{u,v}^{g} = [fe_{rn}, fa_{rn}]$
2	**Output:** R_u^{g+1}
3	Solution Initialization: $R_u: r_{u,v} = lp_v + rand(Up_v - lp_v)$
4	Fitness Evaluation:
5	Fitness function of the proposed ApR-FM2TEDL
6	$Fn(R_{u,v}^{g}) = max\left(Accuracy(R_{u,v}^{g})\right)$
7	Fitness function of the proposed APRMO-SyM model
8	$Fn(R_{u,v}^{g}) = max\left(F1score(R_{u,v}^{g})\right)$
9	if $(g < g_{max}())$
10	{
11	Case (i) Global search phase: if $Nd \geq 0.5$
13	Update the position using. $R_u^{(g+1)} = 0.5(R_{u-1}^{g} + R_u^{g})$
14	Else Case (ii) Local Search: if$Nd < 0.5$
15	{
16	Subcase (i) Unguided Phase: if $HG \leq 0.5$
17	Update the position using. $R_u^{g+1} = R_u^{g} + Ad \times abs(R_{best}^{g} - R_u^{g})$
18	Subcase (ii) Guided Phase: if$HG > 0.5$
19	Find the leader position using $IX_{ldr} = 1 + (IX_u \, mod \, N\, b_{ldr})$
20	Update the position using.$R_u^{g+1} = Ad \times Rand_1 \times cos(2\pi Rand_2)\,(R_{best}^{IX_{ldr}} - R_u^{g}) + R_{best}^{IX_{ldr}}$
21	}
22	}
23	Return $R_{best}^{IX_{ldr}}$
24	End

4.6. Improved BAIT Mitigation Module

The research utilizes an improved BAIT mitigation module that enhances the elimination of potential threats from affecting the network resources in real-time conditions. The conventional approaches mitigate the attacks depending only on the path length and address, whereas the improved version employed in this research considers the time stamp as well. The incorporation of the module improves the identification capability of ApR-FM2TEDL in large-scale network work, where the threat packets are eliminated by deceiving the intruders by focusing on the decoys. When the model identifies the attack data, the BAIT module performs a decoy plan to remove the malicious packets. The decoy plan includes the sending of a Route Request Packet (RREQ) with an address similar to the attack packet, and the malicious node replies with a route reply (RREP). The method verifies the destination address for receiving replies from the nodes that contain route information, and based on the information, the malicious packets are mitigated. The improved BAIT module further updates the information into ApR-FM2TEDL to prevent similar attacks in the future.

5. Results and Discussion

The section examines the simulation results obtained by the ApR-FM2TEDL method with the experimental setup, performance metrics, description of the datasets, and comparative evaluations.

5.1. Experimental Setup

The experimental validation of the ApR-FM2TEDL method for attack detection is carried out in the PyCharm software using the Python programming language. The research utilized a system with 16 GB of RAM that runs on the Windows 10 operating system. The hyperparameters of the proposed network involve the batch size of 128; L2 Regularization, learning rate of 0.001, dropout rate of 0.5, activation function "ReLU", population size of 50, loss function "Binary Cross entropy", and default Optimizer Adam. Further, the above-mentioned experimental setup minimizes the implementation complexities and serves as an effective platform for testing the attack detection models.

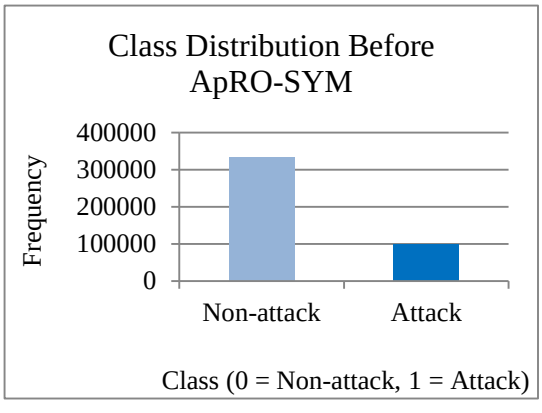
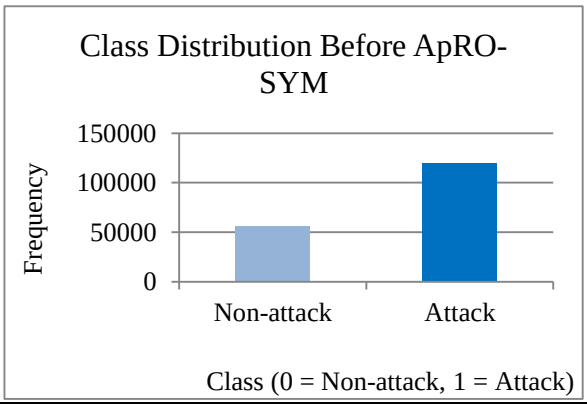
5.2. Dataset Description

CIC-DDoS2019 dataset [33]: The database contains the network traffic data that is specifically utilized for the systems designed for attack detection. The files include the information associated with the internet protocols, as well as various attack types, including both benign and malignant attacks.

BOT-IoT dataset [34]: The dataset comprises network traffic with various file extensions, with the arrangement based on attack and normal categories. The information about the data packets is obtained from the simulated environment and can also be harnessed for attack detection tasks.

5.3. Experimental Outcomes

In this research, the experimental results are shown in Figure 5, which includes the distribution of data over different classes of ApRMO-SyM sampling. The CIC-DDoS2019 database contains attack and non-attack classes having an unequal distribution of 340000 Hz and 10000 Hz, as shown in (a). The inclusion of ApRO-SyM reduces the imbalances to acquire a frequency of 340000 Hz in both classes, as in image (c). Similarly, in the Bot-IoT Database, the data distribution is 120000 for the attack class and 50000 in non-attack classes (b), balanced as 120000 in both classes after the application of the ApRO-SyM sampling method (d). Figure 5 (e), (f) demonstrates the plot of the various features with the value for each attribute. The application of the ApRMO algorithm in ApRO-SyM sampling promotes the generation of better samples by focusing on the effective k-neighbor sets.

Methods	CIC-DDoS2019	BoT-IoT Dataset
Before ApRMO-SyM sampling		
	(a)	(b)

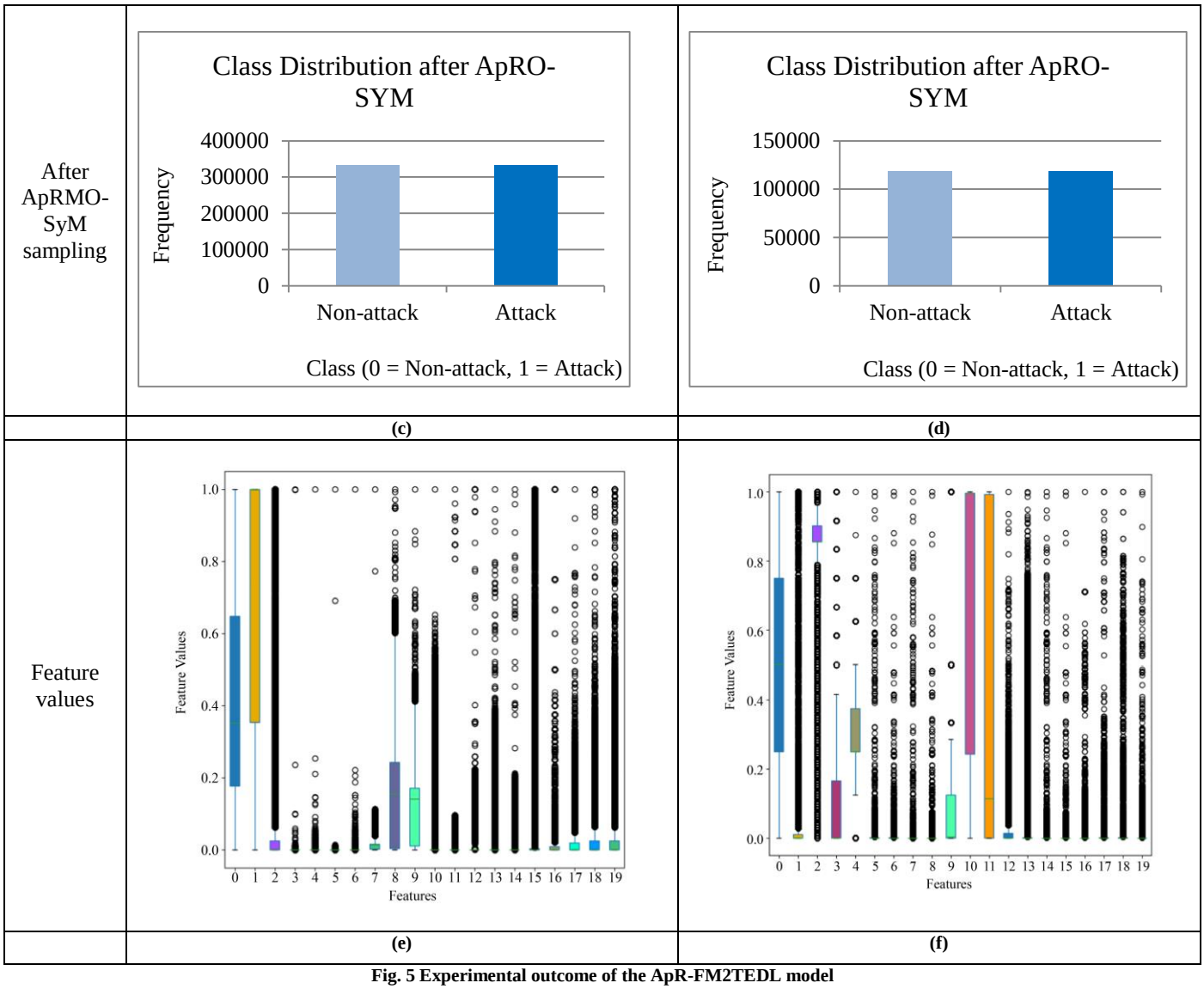


Fig. 5 Experimental outcome of the ApR-FM2TEDL model

5.4. Performance Metrics

The research harnessed the metrics, namely accuracy, specificity, and sensitivity, for computing the ApR-FM2TEDL model's effectiveness in identifying the cloud network attacks.

Generally, the accuracy metric evaluates the correctness of the system in identifying the attacks from the total detections. The original packets that are correctly identified are measured using attacks to sensitivity, and the proportion of real negatives is computed using specificity.

5.5. Performance Analysis

The efficiency of the ApR-FM2TEDL attack detection system is computed using various accuracy, specificity, and sensitivity for K-Fold and training percentage values. The evaluation of the model is interpreted in the CIC-DDoS2019 and BoT-IoT databases, which are discussed in this section.

5.5.1. Performance Analysis for TP and K-Fold using CIC-DDoS2019 Dataset

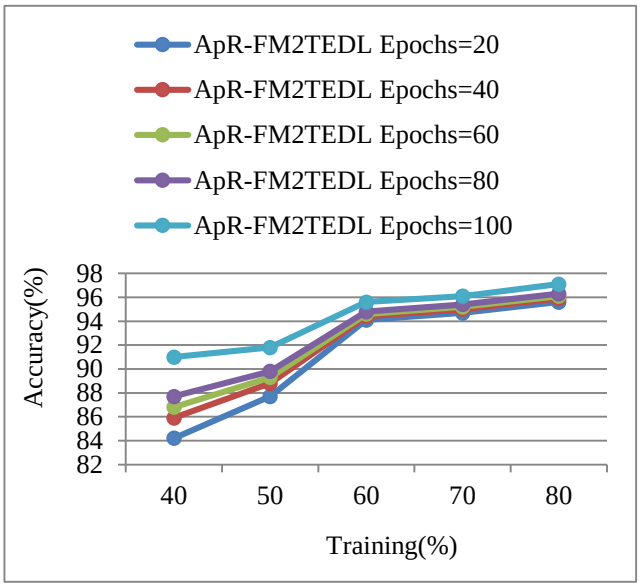
The efficiency of the ApR-FM2TEDL framework in CIC-DDoS2019 is provided in Figure 6 and Table 1. In the training, 70%, the accuracy of ApR-FM2TEDL is 96.12% in epoch 100, and for 80% training, the accuracy is noticed as 97.13%. The sensitivities of ApR-FM2TEDL for varying epochs of 70 and 80 are 97.49% and 98.56%. The ApR-FM2TEDL acquires 96.72% in terms of specificity. Furthermore, with K-fold values of 7 to 10, the performance is measured to attain 95.63% for accuracy in K-fold of 10. The specificity and sensitivity achieve 94.57% and 91.85% in K-fold 8, which improves to 96.99% and 95.20% in K-fold 9. The utilization of the ApRO algorithm minimizes the complexity and results in improvements in detection, as well as the reduction of overfitting issues. Using ApRO-SyM sampling boosts the efficiency. Through the Analysis, it is evident that the performance improves with the increasing number of epochs.

Table 1. Performance analysis using the CIC-DDoS2019 database

Analysis	Training percentage 80			K-fold 10		
	Accuracy (%)	Sensitivity (%)	Specificity (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)
ApR-FM2TEDL (Epochs=20)	95.55	96.77	93.04	92.33	93.46	89.83
ApR-FM2TEDL (Epochs=40)	95.88	97.15	93.98	93.01	94.18	91.04
ApR-FM2TEDL (Epochs=60)	96.08	97.36	94.66	93.34	94.57	91.85
ApR-FM2TEDL (Epochs=80)	96.24	97.66	95.35	93.69	94.97	92.76
ApR-FM2TEDL (Epochs=100)	97.13	98.56	96.72	95.63	96.99	95.20

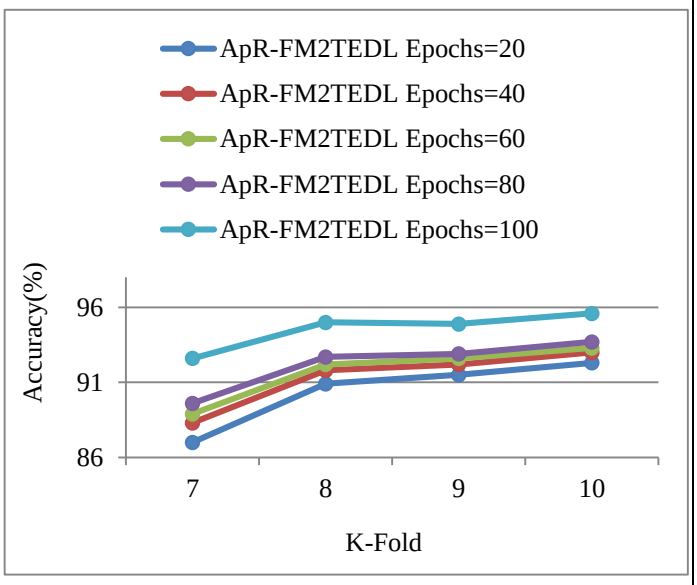
CIC-DDoS2019

TP

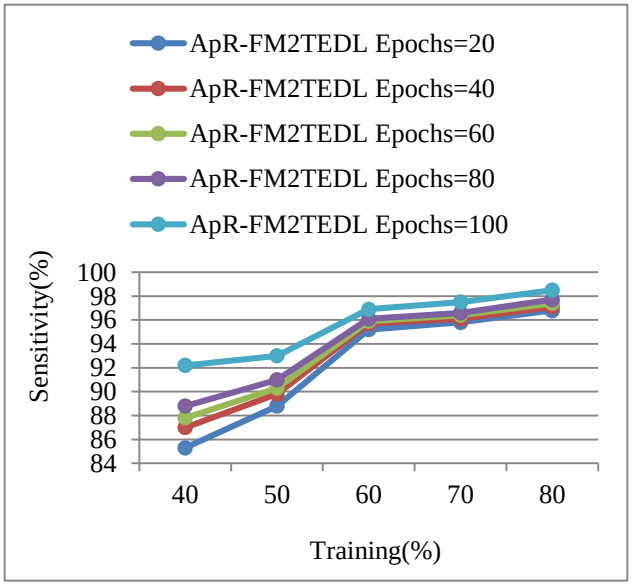


Accuracy

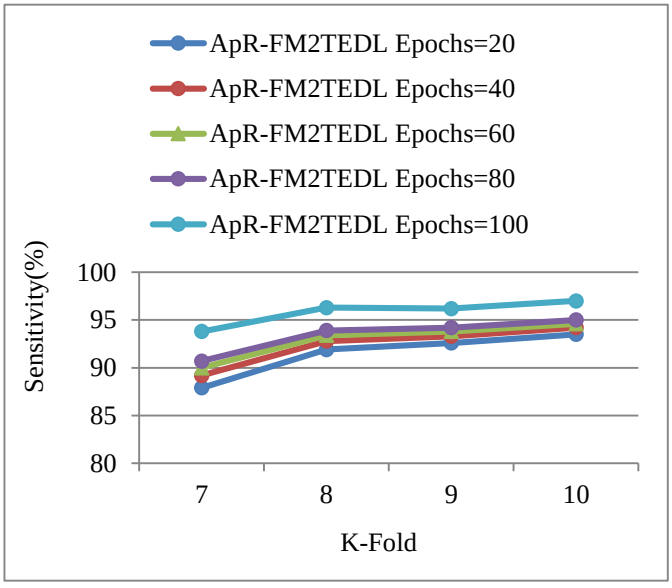
K-Fold



Accuracy



Sensitivity



Sensitivity

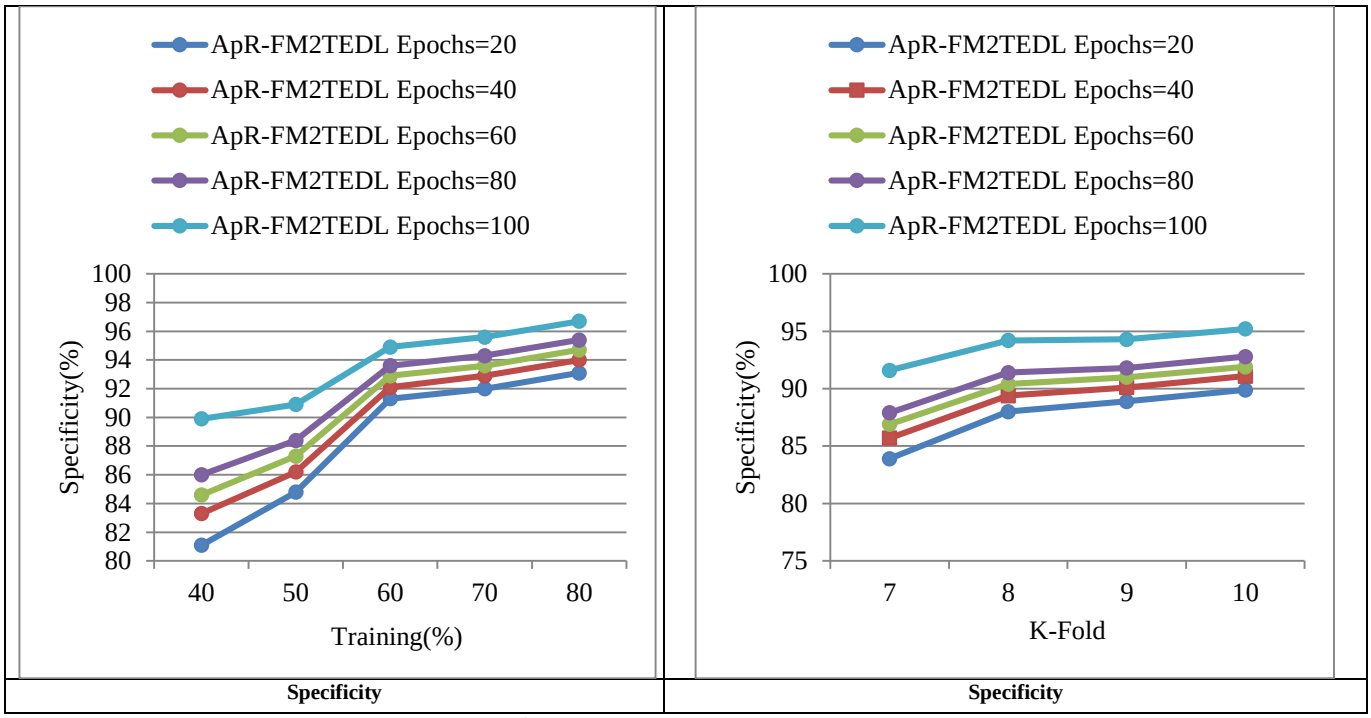


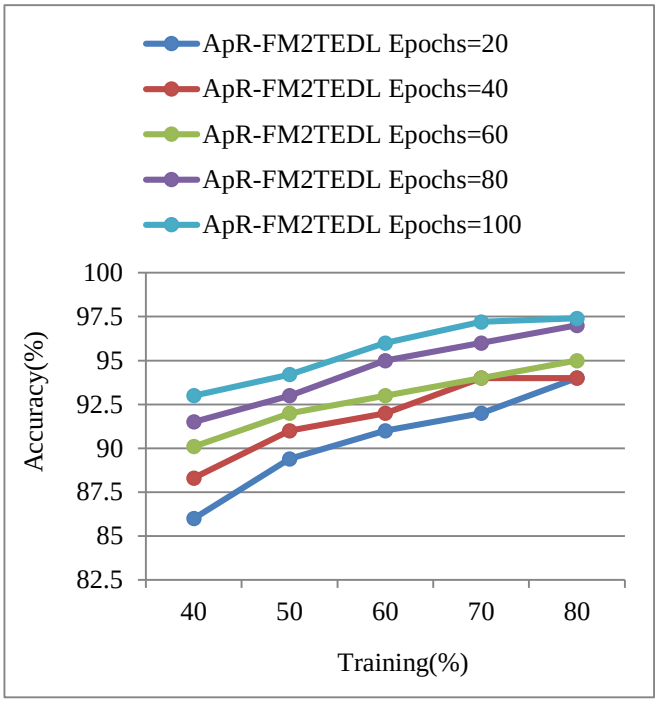
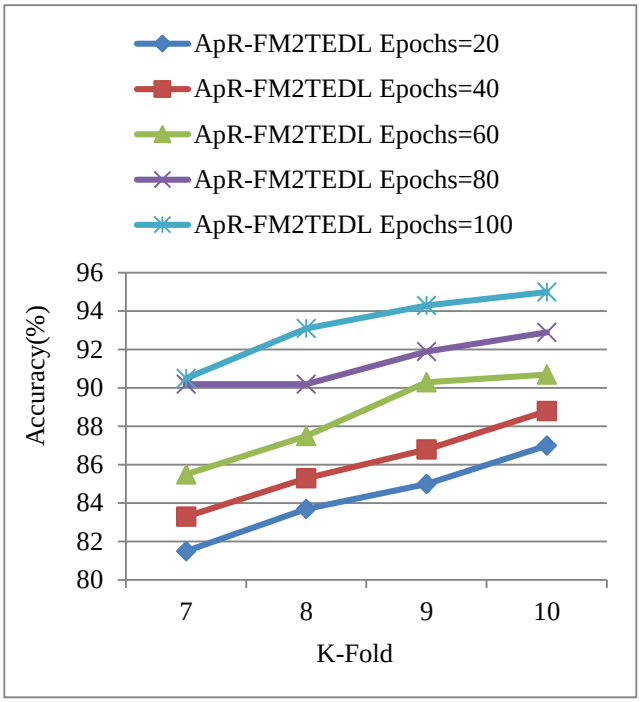
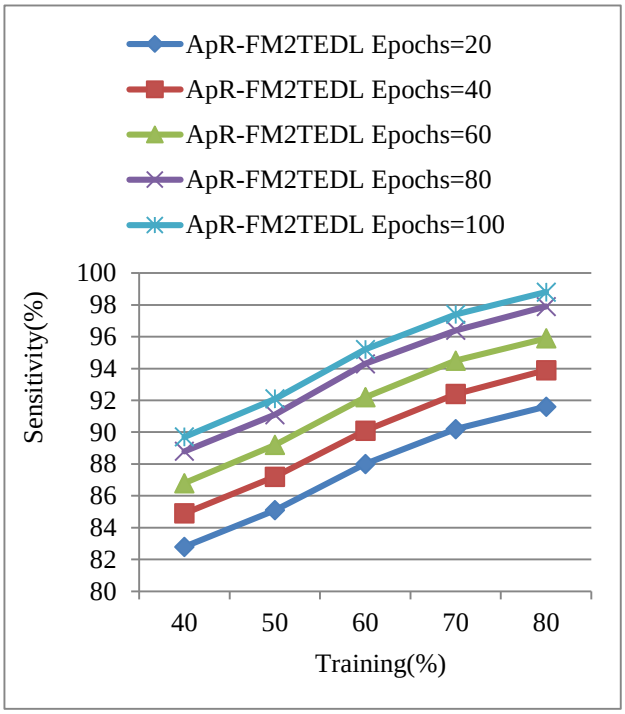
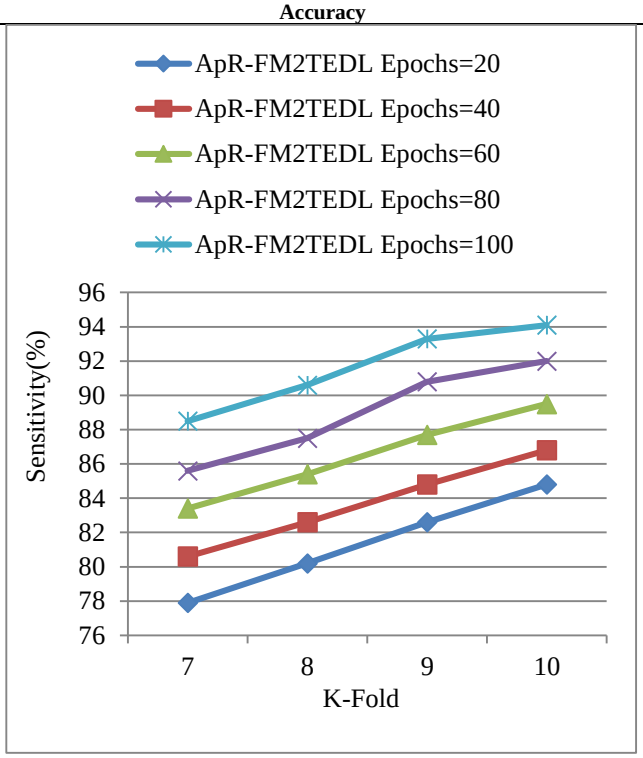
Fig. 6 Performance analysis in CIC-DDoS2019 database

5.5.2. Performance analysis for TP and K-Fold using BOT-IoT database

The performance of ApR-FM2TEDL using the BOT-IoT database is shown in Table 2 and Figure 7 with K-fold as well as training percentage. The accuracy and sensitivity of 97.36% and 96.20% are attained in epoch 100 with 80% training. The specificity acquires 97.83% in 60% of training and improves to 98.59% for 80% of training. Using K-fold 10, the performance is computed, which shows 93.97% accuracy with epoch 40, which improves to95.76% in epoch 100. The values of 94.95% and 96.56% are attained by the ApR-FM2TEDL framework in epoch 100 using sensitivity and specificity measures. Furthermore, the input data quality is improved with the KNN imputation that replaces the missing values with better integers, which reduces the complexities. The ApR-FM2TEDL comprises three DL layers that work in parallel for understanding the complex relationships, paving the way for effective attack detection.

Table 2. Performance analysis using the BOT-IoT database

Analysis	Training percentage 80			K-fold 10		
	Accuracy (%)	Sensitivity (%)	Specificity (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)
ApR-FM2TEDL (Epochs=20)	90.61	91.17	96.87	93.17	93.48	92.79
ApR-FM2TEDL (Epochs=40)	92.36	93.17	97.21	93.97	93.63	94.21
ApR-FM2TEDL (Epochs=60)	93.38	94.73	97.83	94.87	93.87	95.78
ApR-FM2TEDL (Epochs=80)	96.76	95.48	98.17	95.38	94.17	96.50
ApR-FM2TEDL (Epochs=100)	97.36	96.20	98.59	95.76	94.95	96.56

<table>
<tr><th colspan="2">BOT-IoT database</th></tr>
<tr><th>TP</th><th>K-Fold</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><th>Accuracy</th><th>Accuracy</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><th>Sensitivity</th><th>Sensitivity</th></tr>
</table>

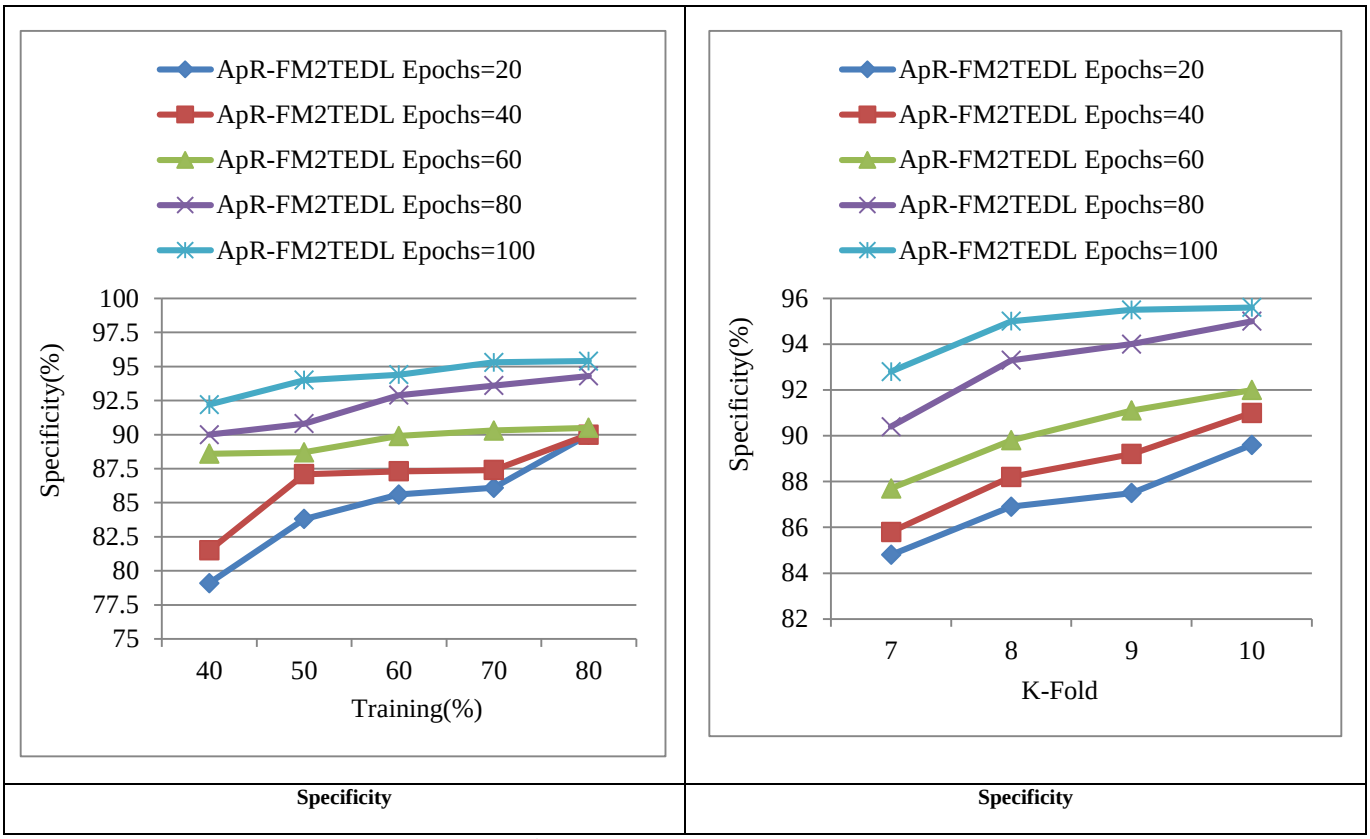


Fig. 7 Performance analysis in the BOT-IoT database

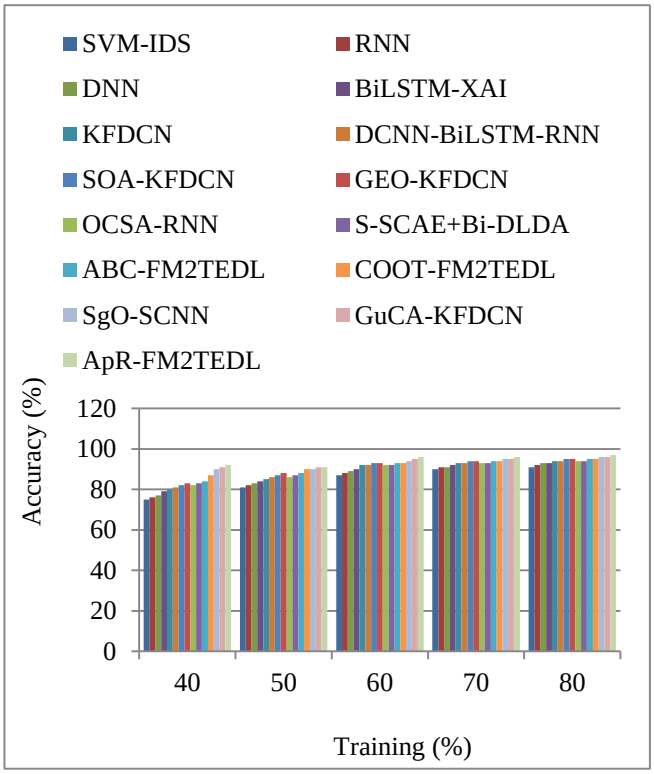
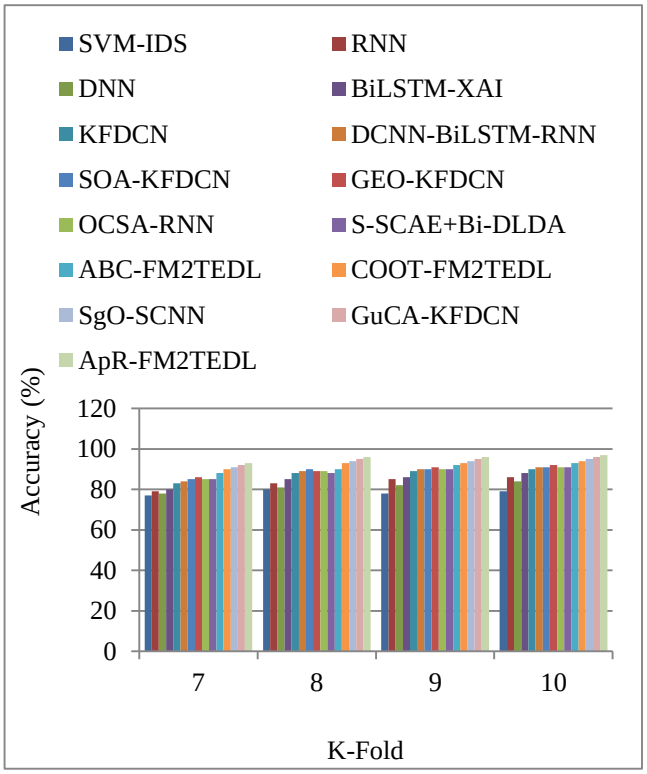
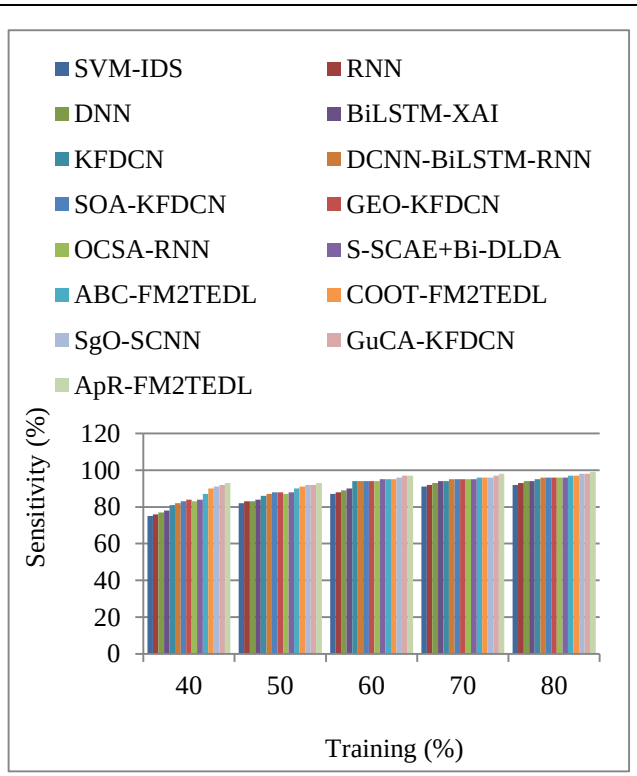
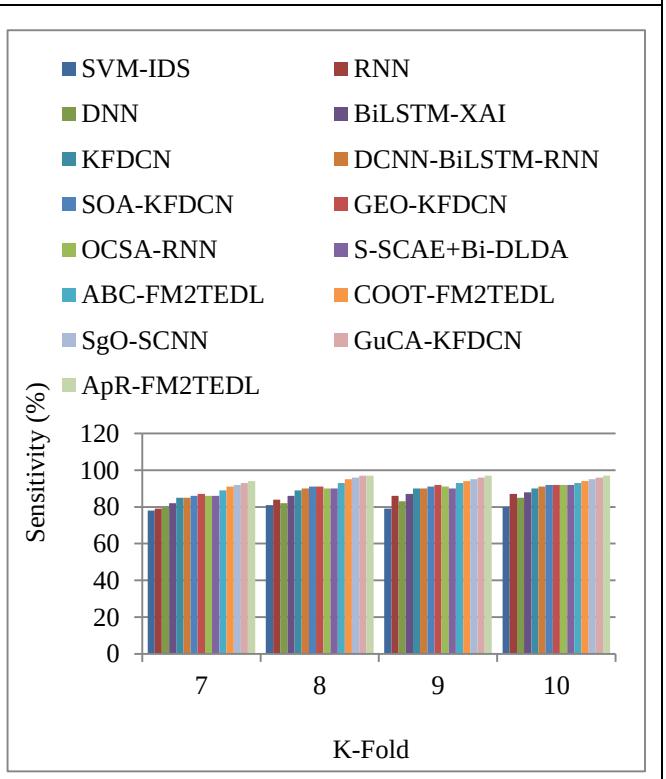
5.6. Comparative Analysis

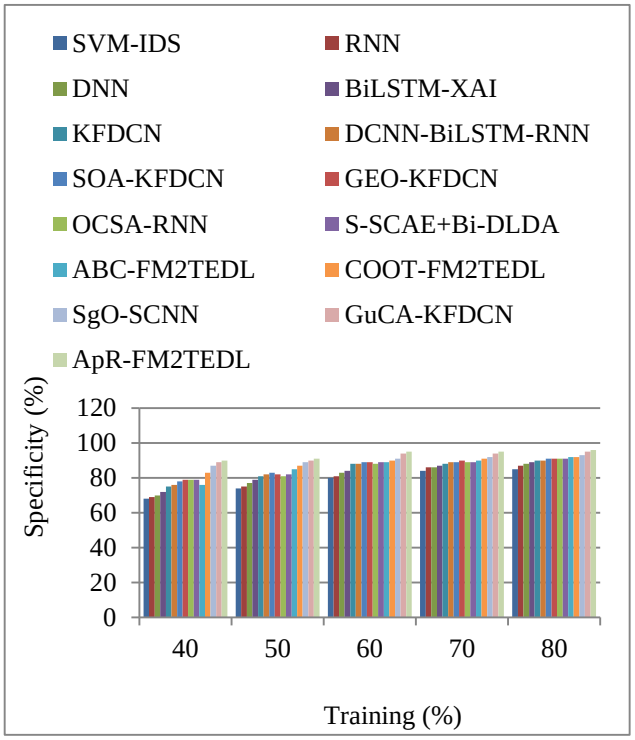
The outcome of ApR-FM2TEDL in detecting the attacks are compared against the methods namely Recurrent Neural Network (RNN) [30], Support Vector Machine for Intrusion Detection System (SVM-IDS) [35], Deep Neural Network (DNN) [36], Bidirectional LSTM enabled explainable Artificial Intelligence (BiLSTM-XAI) [37], KFDCN, DCNN-BiLSTM-RNN [38], Seagull optimization enabled KFDCN (SOA-KFDCN) [39], Oppositional Crow Search Algorithm based Recurrent Neural Network (OCSA-RNN) [40], Sparse autoencoder and stacked contractive autoencoder along with a Bi-DLDA (Bi-directional LSTM with attention mechanism) (S-SCAE + Bi-DLDA) [41], Golden Eagle Optimized KFDCN (GEO-KFDCN) [42], Artificial Bee Colony enabled FM2TEDL (ABC-FM2TEDL) [32], COOT-FM2TEDL [31], Seagull optimized SMOTE enabled DCNN (SgO-SCNN) [43], and GuCA-KFDCN. The experimental outcomes of ApR-FM2TEDL in comparison with the mentioned approaches are detailed in this section.

5.6.1. Comparative Analysis for Training Percentage and K-Fold using CIC-DDoS2019 dataset

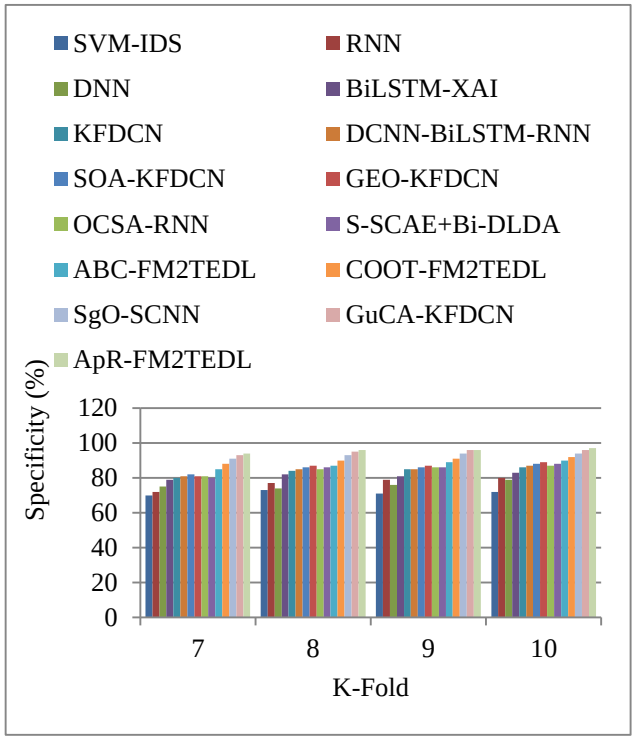
The effectiveness of the attack detection ApR-FM2TEDL model is compared against the prevailing methods, which are shown in Figure 8. With training 80%, the proposed method shows an accuracy of 97.13%, surpassing the existing techniques in attack detection with the significant improvement of 5.73% over SVM-IDS,4.67% over RNN,5.20% over DNN,4.15% over BiLSTM-XAI,2.98% over KFDCN, 2.83% over DCNN-BiLSTM-RNN, 2.67% over SOA-KFDCN, 2.36% over GEO-KFDCN, 2.74% over OCSA-RNN, 2.57% over S-SCAE+Bi-DLDA, 2.75% over ABC-FM2TEDL,1.74% over COOT-FM2TEDL, and 1.1% over SgO-SCNN The specificity of ApR-FM2TEDL shows greater outcomes with 96.72%, while the ABC-FM2TEDL shows a lower value with 90.35%. The ApR-FM2TEDL attains a higher sensitivity as 98.56%, which outperforms SVM-IDS and SgO-SCNN by 6.30% and 1.38%.

Through the K-fold value of 10, the ApR-FM2TEDL, the accuracy measures 95.63%; however, the GEO-KFDCN and DCNN-BiLSTM-RNN reveal 4.21% and 5.19%. The sensitivity as well as specificity in ApR-FM2TEDL showcases 96.99% and 95.20%, which is higher than COOT-FM2TEDL, which shows 93.91% and 89.60%. Specifically, the ApR-FM2TEDL method involves three layers of parallel working DL models that promote the learning of complex features. Besides, the ApRMO algorithm offers effective hyperparameter tuning leading to accurate attack detection results. Compared to the existing techniques, the ApR-FM2TEDL model improved the overall detection performance and shows great potential for the application of attack detection in cloud networks.

TP analysis	K-Fold analysis
	
Accuracy	Accuracy
	
Sensitivity	Sensitivity



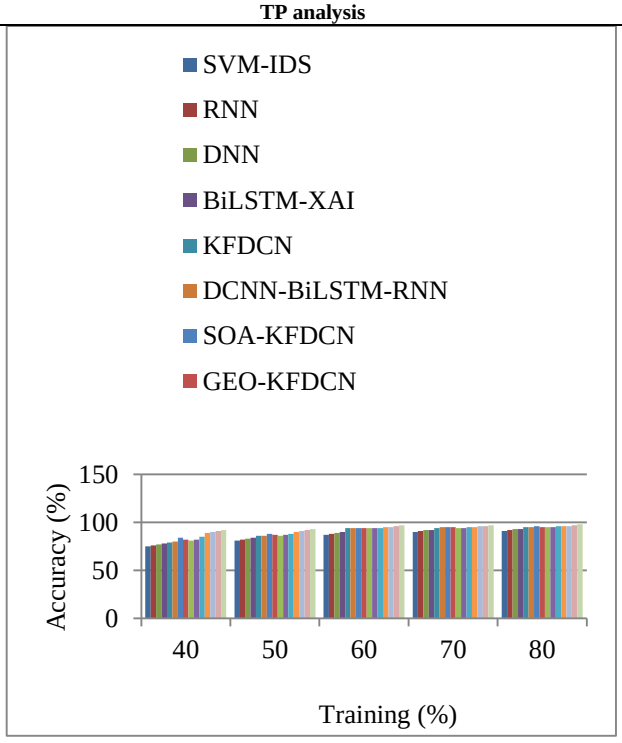
Specificity



Specificity

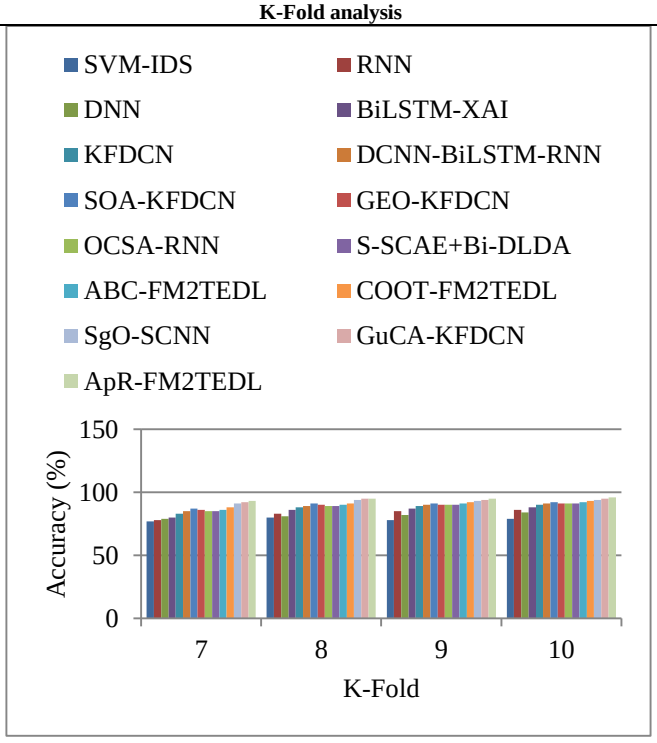
Fig. 8 Comparative analyses using the CIC-DDoS2019 database

TP analysis



Accuracy

K-Fold analysis



Accuracy

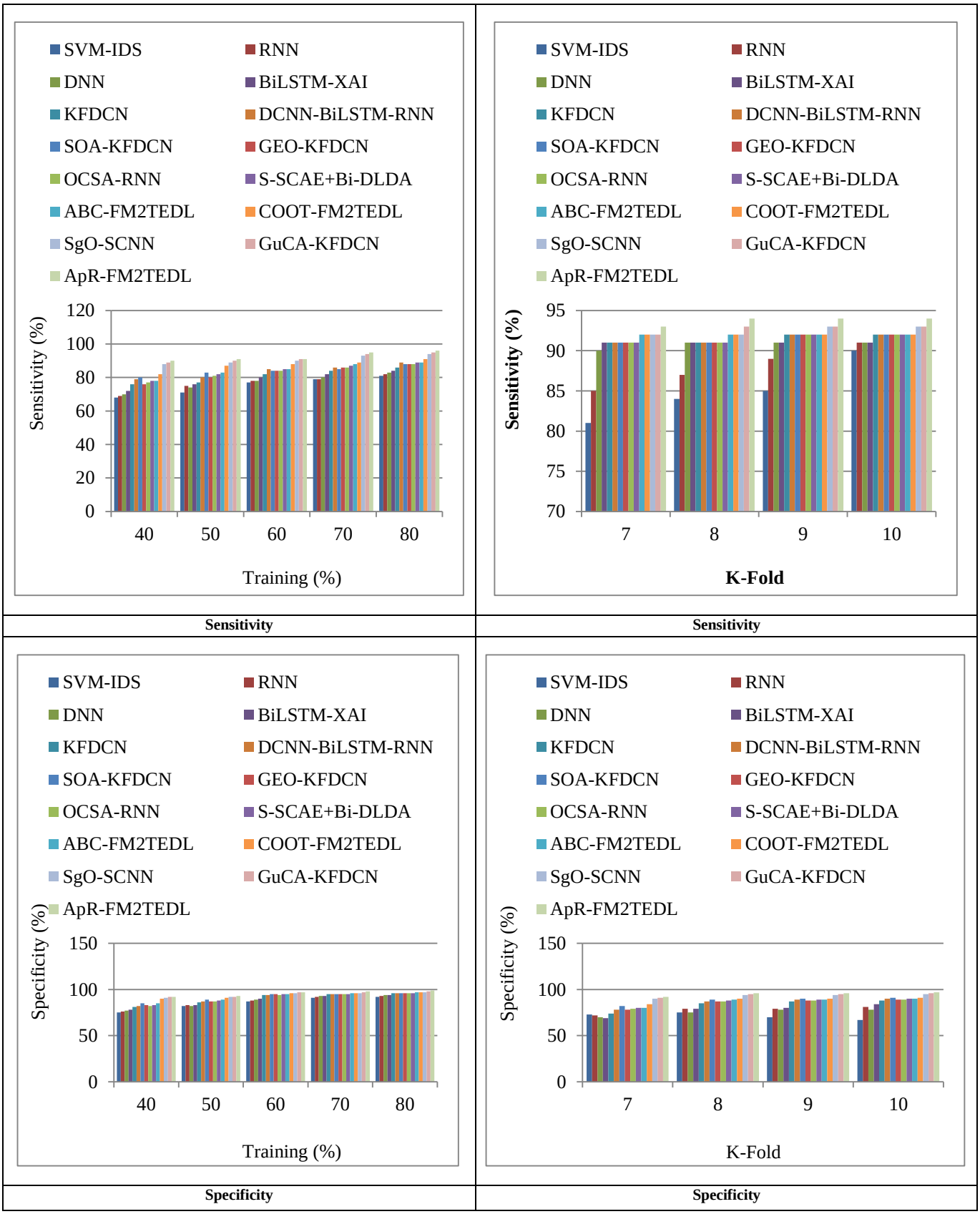


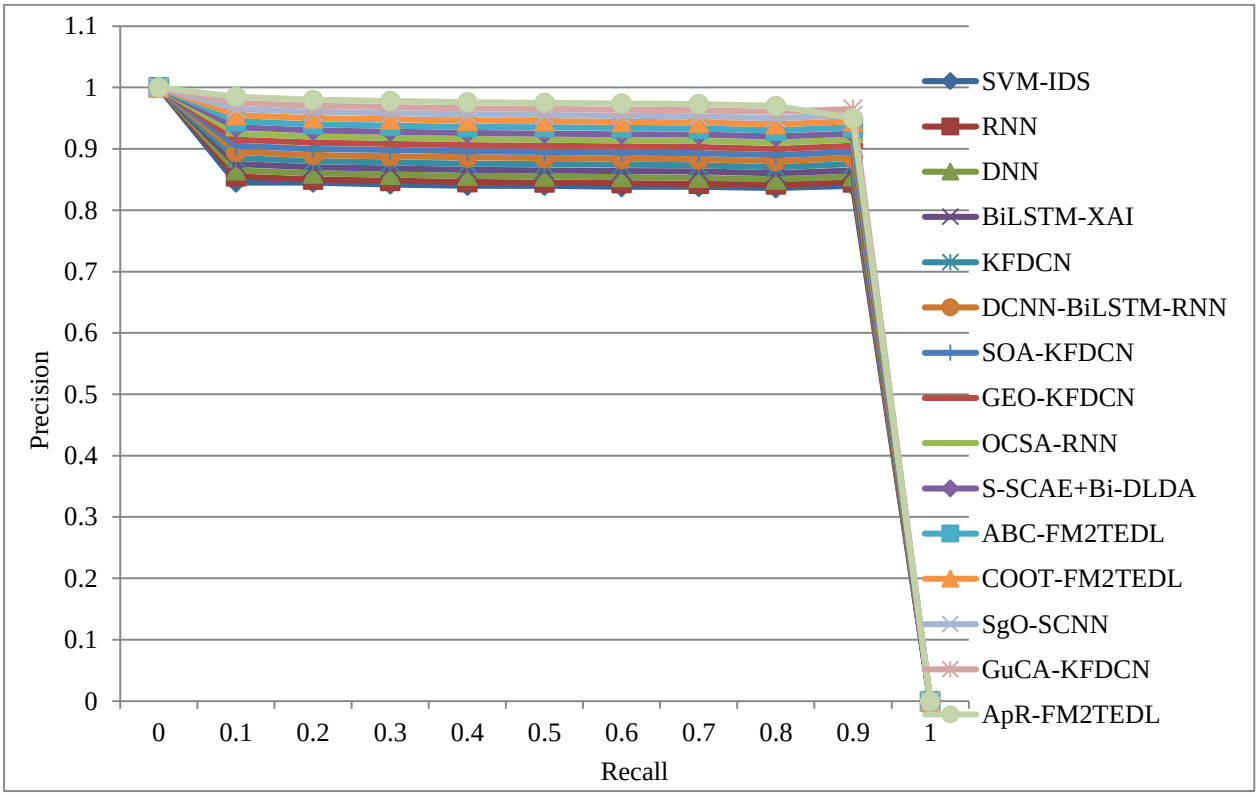
Fig. 9 Comparative analyses using the BOT-IoT database

5.6.2. Comparative Analysis for TP and K-Fold using BOT-IoT database

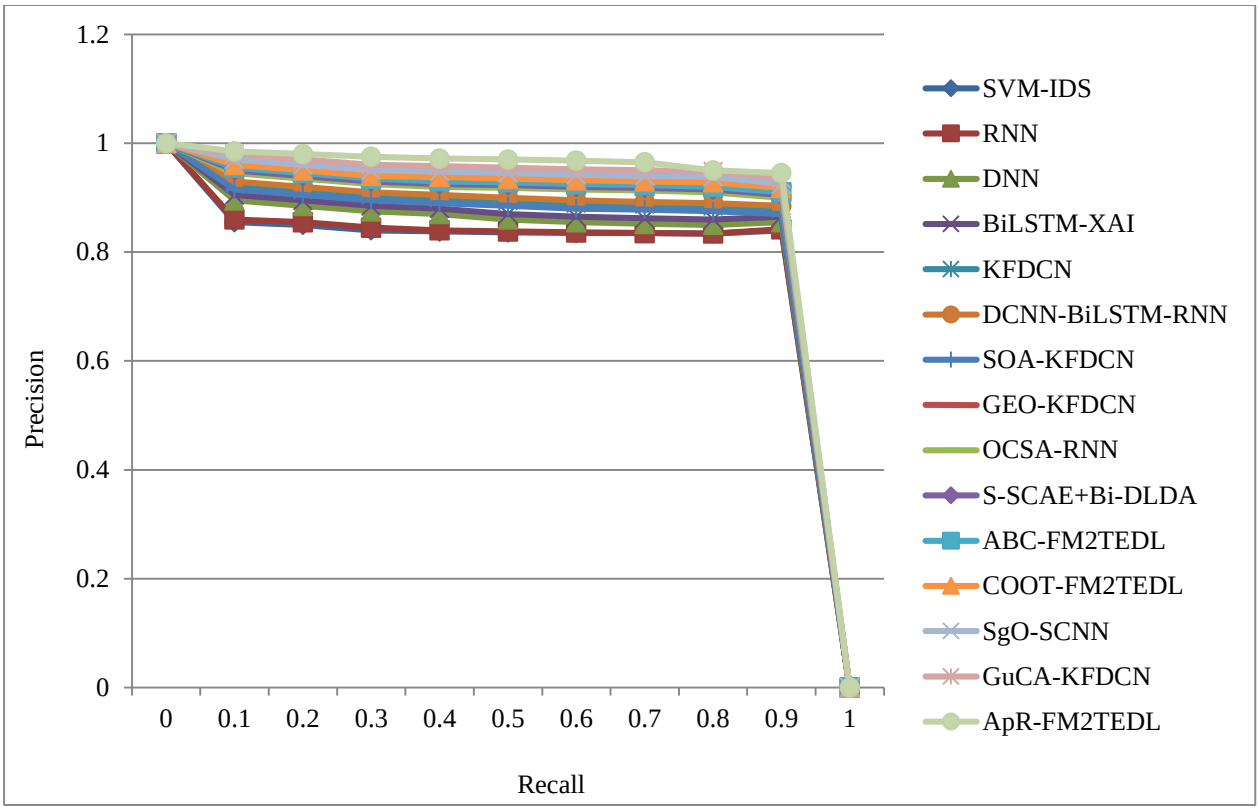
The attack detection efficiency of ApR-FM2TEDL, represented in Figure 9, is compared with the baseline approaches. For a training percentage of 80, the ApR-FM2TEDL method reveals an accuracy of 97.74%, exceeding the other existing models with the relative improvement of 6.32% against SVM-IDS,5.27% against RNN,5.80% against DNN,4.75% against BiLSTM-XAI,2.84% against KFDCN, 2.56% against DCNN-BiLSTM-RNN, 2.29% against SOA-KFDCN, 2.56% against GEO-KFDCN, 2.55% against OCSA-RNN, 2.44% against S-SCAE+Bi-DLDA, 2.43% against ABC-FM2TEDL,2.15% against COOT-FM2TEDL, 1.73% against SgO-SCNN, and 0.92% against GuCA-KFDCN. The specificity of ApR-FM2TEDL shows greater outcomes with 98.59%, while the ABC-FM2TEDL shows a lower value with 96.38%. The ApR-FM2TEDL attains a higher sensitivity of 96.20%, outperforming the existing SVM-IDS and SgO-SCNN by 15.41% and 1.79%, respectively. With a K-fold value of 10, the ApR-FM2TEDL, the accuracy measures 95.75%; however, the GEO-KFDCN and COOT-FM2TEDL reveal 91.36% and 92.72 %. The sensitivity as well as specificity in ApR-FM2TEDL showcases 94.95% and 96.56%, which is higher than COOT-FM2TEDL, which shows 93.22% and 92.21%. The imbalanced classes in the datasets are balanced using the ApRO-SyM sampling technique during the generation of samples. Moreover, the ApRMO algorithm assists in choosing the better K-neighbors for standard synthetic data that boosts the detection efficiency. Further, the ApR-FM2TEDL method involves Tri-ensemble DL models offering the high feature representation ability to learn the complex malicious patterns present in the network traffic. Moreover, the ApRMO algorithm optimally tunes the hyperparameters by improving the process of training, resulting in improved detection performance. From this, the proposed technique demonstrates impressive outcomes with performance metrics compared to the other existing models.

5.7. PRC Analysis

In this section, the Precision-Recall Curve (PRC) analysis of the ApR-FM2TEDL framework is validated against the conventional approaches as represented in Figure 10. Using the CIC-DDoS2019 database in recall 0.8, the ApR-FM2TEDL acquires a precision value of 0.83, which improves KFDCN and DNN, which acquire only 0.77 and 0.76. The precision at 0.1 shows 0.96, while the DCNN-BiLSTM-RNN exhibits only 0.94. Similarly, the Analysis in the BOT-IoT database shows a higher precision of 0.96, while the SVM-IDS and ABC-FM2TEDL reveal 0.91 and 0.94, respectively. The deeper layers of the ApR-FM2TEDL attack detection framework captured the intrinsic features of the threat patterns in the data that pave the way for accurate detection. Furthermore, data undergoes missing imputation using the KNN model that minimizes the complexity, leading to higher precision.



CIC-DDoS2019 dataset



BOT-IoT dataset

Fig. 10 PRC Analysis

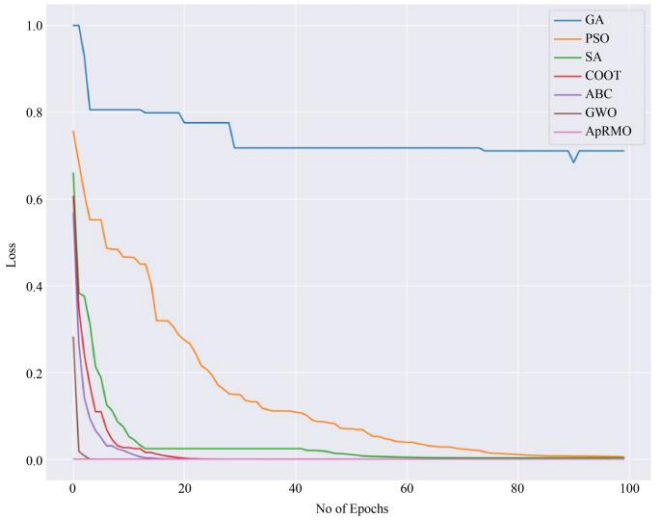


Fig. 11 Convergence Analysis

5.8. Convergence Analysis

The convergence analysis of the proposed ApR-FM2TEDL is carried out for validating different attack detection algorithms in reaching the optimal solution, and the results are shown in Figure 11. For iteration 100, the existing standard optimization GA is reported with a loss of 0.710, while the PSO obtains 0.0049, SA obtains 0.003, COOT obtains 4.04E-09, ABC obtains 2.82E-17, and GWO obtains a loss of 5.96E-56. Despite the fact that challenges faced by the conventional optimization methods like the GA, PSO, GWO, and COOT are described as they fail to handle complex data and have limited performance. Additionally, the ApRMO algorithm uses random search and efficiently traverses the large search space, as well as provides equal priority during exploitation and exploration, which results in high convergence speed for reaching the optimal solution. Ultimately, the ApRMO algorithm attains the optimal solution and reduces the loss by overcoming the other existing optimization algorithms.

5.9. Ablation Study

An ablation study is conducted to validate the significance of the several components involved in designing the proposed ApR-FM2TEDL model for attack detection. Specifically, the ablation study evaluates the significance of each component, including the ApRMO, ApRO-SyM, and the tri-ensemble model, by removing the component each time and thus evaluates the contribution of the specific components in improving the overall detection performance. From the ablation study, it is reported that the proposed ApR-FM2TEDL achieves an accuracy of 97.74%, surpassing the proposed method without ApR-SyM by 4.20%, FM2TEDL without ApRMO algorithm by 3.18%, and ApR-TEDL without Fuzzy min-max approach by 1.35%. Moreover, the proposed ApR-FM2TEDL model, combining the strength of ApRMO,

ApRO-SyM, the fuzzy min-max approach, and the Tri-ensemble model, obtained high detection accuracy. In the proposed framework, the ApRO-SyM sampling method generates the optimal synthetic samples over the minority class that eliminates the overfitting issues. Moreover, the ApRMO algorithm carries out the effective parameter tuning of the classifier and improves the ApRO-SyM sampling for data generation by selecting the better neighbor sets. Besides, the Tri-ensemble Deep learning method offers a high feature learning ability to detect the attack patterns present in network traffic, and the extracted features are processed with the element-wise min-max operation, where the maximum and minimum data features are acquired separately, and facilitate the effective detection in cloud computing. Moreover, the application of the specific components together facilitates improving the overall detection accuracy, as is evident from the ablation study. Figure 12 portrays the ablation study of the proposed method for attack detection.

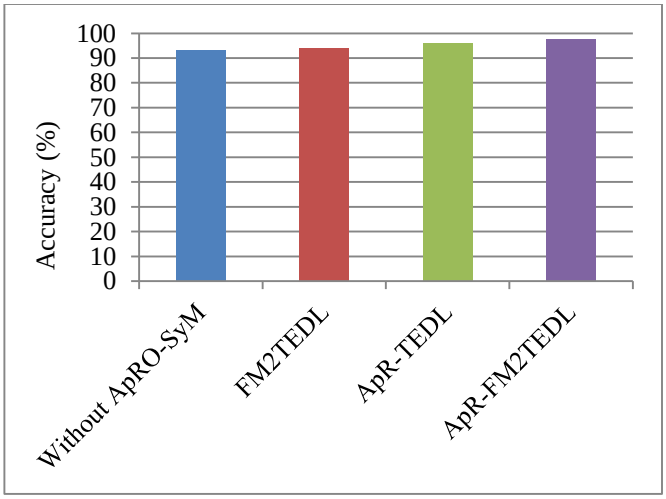


Fig. 12 Ablation Study

5.10. Computation Complexity Analysis

Computation complexity analysis is conducted to evaluate the computation efficiency of the different methods designed for attack detection in cloud networks. Figure 13 visualizes the computational outcomes acquired for the model. In this Analysis, the ApR-FM2TEDL method takes a low computation time of 0.01 s for the attack detection. Contrastingly, the baseline technique SVM-IDS takes the computation time of 0.05s, RNN takes 0.06s, DNN takes 0.05s, BiLSTM-XAI takes 0.08s, KFDCN takes 0.06s, DCNN-BiLSTM-RNN takes 0.06s, SOA-KFDCN takes 0.05s, GEO-KFDCN takes 0.04s, OCSA-RNN takes 0.04s, S-SCAE+Bi-DLDA takes 0.03s, ABC-FM2TEDL takes 0.02s, COOT-FM2TEDL takes 0.04s, SgO-SCNN takes 0.03s, and GuCA-KFDCN takes 0.02s Moreover, the proposed ApR-FM2TEDL overcomes the other state-of-the-art techniques in terms of computation efficiency by significantly reducing the computational complexity. The application of the ApRMO algorithm provides the optimal configuration to reduce the training time, which in turn minimizes the computation complexity of the proposed model.

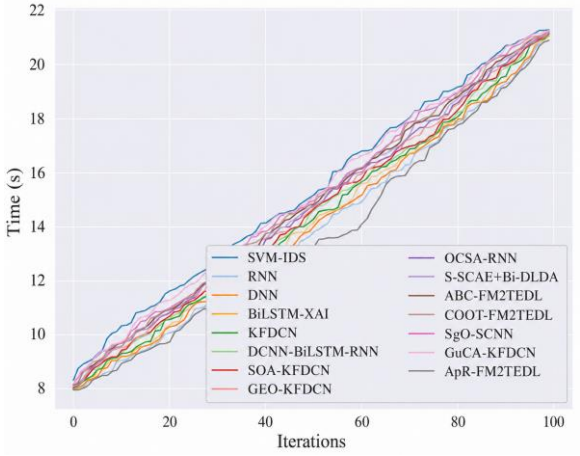


Fig. 13 Computation complexity Analysis

5.11. Comparative Evaluation of Different Optimization Algorithms

The section analyzes the comparative evaluation of the ApRMO algorithm with different existing optimization algorithms, such as the PSO, GA, SA, COOT, ABC, and GWO algorithms. Based on the comparative evaluation, the proposed ApRMO algorithm achieves a high accuracy of 97.74%, exceeding the existing algorithm GA by 0.92%, PSO by 3.47%, SA by 1.73%, COOT by 2.15%, ABC by 2.43%, and GWO by 0.91%. Compared with existing optimization algorithms, the proposed ApRMO algorithm demonstrates superior performance by reaching the global optimal solution. In contrast, conventional optimization algorithms, including the GA, PSO, SA, COOT, ABC, and GWO, are limited in terms of slow convergence during the local search phases, limiting the overall accuracy. In addition, the existing algorithms face difficulty while addressing the complex and high-dimensional environments. Subsequently, the proposed ApRMO eliminates the drawbacks regarding the complexities, convergence, and local optima trapping and reaches the global optimal solution, resulting in improving the overall accuracy. Figure 14 depicts the Comparison of different Optimization Algorithms.

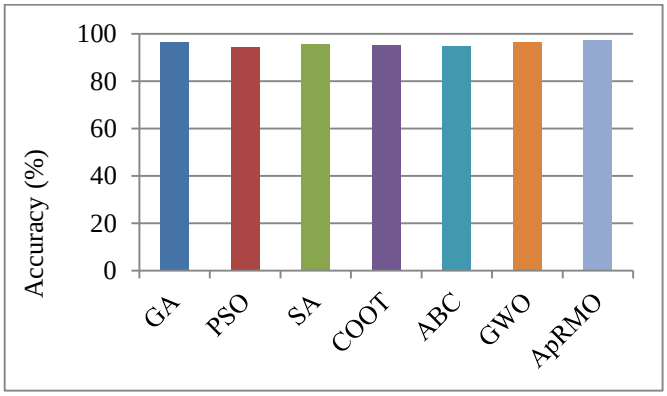


Fig. 14 Comparison of different Optimization Algorithms

5.12. Comparative Discussion

The results reveal the superior performance of ApR-FM2TEDL when compared to the conventional approaches that are denoted in Tables 3 and 4. The utilization of the developed ApRO-SyM sampling approach eliminated the overfitting issues where effective neighbor sets were selected for generating the samples. The incorporation of the technique enhanced the detection efficiency, while the KFDCN and DNN exhibited biased outcomes due to the higher imbalance issues. The lack of optimization in SVM-IDS and DCNN-BiLSTM-RNN increased the complexity of detection, where the researchers overcame the limitations in ApR-FM2TEDL with the development of the ApRMO algorithm. The ApRMO tunes the hyperparameters as well as supports the neighbor set estimation during the sample generation process, leading to standardized production of synthetic data. The COOT-FM2TEDL and GuCA-KFDCN show lower accuracy in detection than the ApR-FM2TEDL, which identifies the complex behavior of attack packets, and the fusion technique interpreted both the maximum and minimum instances. Moreover, the proposed ApR-FM2TEDL scheme exhibits superior performance and can be employed to detect attacks in cloud environments.

Table 3. Comparative discussion of the ApR-FM2TEDL framework in CIC-DDoS2019

CIC-DDoS2019 dataset						
Methods	**TP 80**			**K-Fold 10**		
	Accuracy (%)	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**	**Sensitivity (%)**	**Specificity (%)**
SVM-IDS [35]	91.56	92.35	85.07	78.82	79.50	72.31
RNN [30]	92.59	93.43	86.60	86.22	86.98	80.22
DNN [36]	92.08	92.97	86.59	84.58	85.39	79.08
BiLSTM-XAI [37]	93.10	94.04	88.11	88.09	88.96	83.09
KFDCN	94.23	95.23	89.74	90.36	91.29	85.87
DCNN-BiLSTM-RNN [38]	94.38	95.41	90.15	90.67	91.63	86.43
SOA-KFDCN [39]	94.53	95.58	90.55	90.98	91.96	86.99
GEO-KFDCN [42]	94.84	95.94	91.35	91.60	92.64	88.11
OCSA-RNN [40]	94.47	95.59	90.43	90.85	92.41	87.13
S-SCAE+ Bi-DLDA [41]	94.63	95.69	91.21	91.53	92.42	87.88
ABC-FM2TEDL [32]	94.46	95.49	90.35	90.82	91.79	86.71
COOT-FM2TEDL [31]	95.44	96.57	92.21	92.84	93.91	89.60
SgO-SCNN [43]	96.05	97.20	93.06	94.08	95.18	91.09
GuCA-KFDCN	96.28	97.69	95.80	94.77	96.12	94.28
Ensemble DNN [44]	89.4	-	-	-	-	-
GBT [45]	93.62	-	-	-	-	-
Hybrid Deep Learning model [46]	80.75	-	-	-	-	-
BI-LSTM-GMM [47]	94	-	-	-	-	-
CNN [48]	95.4	-	-	-	-	-
XGBoost [49]	91.26	-	-	-	-	-
ApR-FM2TEDL	97.13	98.56	96.72	95.63	96.99	95.20

Table 4. Comparative discussion of the ApR-FM2TEDL framework in the BoT-IoT Dataset

BOT-IoT dataset						
Methods	**TP 80**			**K-Fold 10**		
	Accuracy (%)	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**	**Sensitivity (%)**	**Specificity (%)**
SVM-IDS [35]	91.56	81.37	92.20	78.82	90.82	66.81
RNN [30]	92.59	82.40	93.23	86.22	91.90	80.53
DNN [36]	92.08	81.89	92.73	84.58	91.40	77.76
BiLSTM-XAI [37]	93.10	82.91	93.78	88.09	92.40	83.78
KFDCN	94.97	84.77	95.66	90.46	92.93	87.98
DCNN-BiLSTM-RNN [38]	95.24	87.19	95.94	91.36	93.05	89.68

SOA-KFDCN [39]	95.51	89.62	96.22	92.27	93.17	91.37
GEO-KFDCN [42]	95.24	87.19	95.96	91.36	93.05	89.68
OCSA-RNN [40]	95.25	88.32	96.03	91.54	93.07	90.09
S-SCAE+ Bi-DLDA [41]	95.36	88.40	96.05	91.54	93.09	90.37
ABC-FM2TEDL [32]	95.37	88.41	96.08	91.82	93.11	90.53
COOT-FM2TEDL [31]	95.64	90.84	96.38	92.72	93.22	92.22
SgO-SCNN [43]	96.05	94.48	96.79	94.08	93.40	94.76
GuCA-KFDCN	96.84	95.29	97.67	94.83	94.02	95.63
TEHO-DBN [50]	83.0	-	-	-	-	-
SD-LVQ [51]	90.2	-	-	-	-	-
GHLBO-based DSA [3]	91.70	-	-	-	-	-
PB-DID [52]	96.3	-	-	-	-	-
ApR-FM2TEDL	97.74	96.20	98.59	95.76	94.95	96.56

5.13. Statistical Analysis

Statistical significance testing was obtained in terms of different statistical measures to evaluate the robustness of the model. The statistical measures, including the mean, variance, and Standard deviation (STD), are utilized for evaluating the results reported in terms of metric accuracy, sensitivity, and specificity. Further, the overall statistical results depicted in Tables 5 and 6 show that the ApR-FM2TEDL framework outperforms other techniques utilized for the comparison by achieving highly efficient results.

Table 5. Statistical Analysis with the CIC-DDoS2019 dataset with TP Analysis

Methods	**TP 80**								
	Accuracy			**Sensitivity**			**Specificity**		
	Mean	Variance	STD	Mean	Variance	STD	Mean	Variance	STD
SVM-IDS	85.22	38.66	6.22	85.88	39.84	6.31	78.32	42.22	6.50
RNN	86.26	38.77	6.23	86.97	39.96	6.32	79.86	42.34	6.51
DNN	85.74	38.77	6.23	86.51	39.96	6.32	79.85	42.34	6.51
BiLSTM-XAI	86.77	38.80	6.23	87.58	39.98	6.32	81.37	42.37	6.51
KFDCN	89.23	29.87	5.47	90.11	30.85	5.55	84.34	32.89	5.74
DCNN-BiLSTM-RNN	89.58	27.07	5.20	90.48	27.99	5.29	84.93	29.95	5.47
SOA-KFDCN	89.92	24.41	4.94	90.85	25.27	5.03	85.53	27.15	5.21
GEO-KFDCN	90.61	19.51	4.42	91.59	20.27	4.50	86.72	21.98	4.69
OCSA-RNN	90.07	22.37	4.73	91.07	23.84	4.88	86.03	21.98	4.69
S-SCAE+ Bi-DLDA	90.45	19.18	4.38	91.32	21.57	4.64	86.39	23.75	4.87
ABC-FM2TEDL	89.75	25.72	5.07	90.67	26.61	5.16	85.23	28.53	5.34
COOT-FM2TEDL	91.99	11.43	3.38	93.00	11.98	3.46	88.35	13.34	3.65
SgO-SCNN	93.37	5.61	2.37	94.41	5.98	2.45	89.98	6.98	2.64
GuCA-KFDCN	93.51	5.93	2.44	94.80	6.32	2.51	92.62	7.34	2.71
ApR-FM2TEDL	94.32	6.08	2.47	95.62	6.50	2.55	93.53	7.39	2.72

Table 6. Statistical Analysis with Bot-IoT Dataset with TP Analysis

Methods/ Metrics	**TP 80**								
	Accuracy			**Sensitivity**			**Specificity**		
	Mean	Variance	STD	Mean	Variance	STD	Mean	Variance	STD
SVM-IDS	85.22	38.66	6.22	75.41	23.68	4.87	85.82	38.82	6.23
RNN	86.26	38.77	6.23	76.93	20.48	4.53	86.87	39.03	6.25
DNN	85.74	38.77	6.23	76.42	20.48	4.53	86.38	38.96	6.24
BiLSTM-XAI	86.77	38.80	6.23	77.44	20.48	4.53	87.42	39.02	6.25
KFDCN	89.19	42.50	6.52	79.60	21.92	4.68	89.87	42.63	6.53

DCNN-BiLSTM-RNN	90.24	29.87	5.47	82.37	16.72	4.09	90.93	29.99	5.48
SOA-KFDCN	91.28	19.51	4.42	85.13	12.56	3.54	91.98	19.62	4.43
GEO-KFDCN	90.24	29.87	5.47	82.37	16.72	4.09	90.95	29.96	5.47
OCSA-RNN	90.28	29.51	5.43	82.89	17.46	4.18	91.04	28.87	5.37
S-SCAE+ Bi-DLDA	90.50	28.17	5.31	83.50	15.76	3.97	91.30	26.10	5.11
ABC-FM2TEDL	90.76	24.41	4.94	83.75	14.51	3.81	91.45	24.52	4.95
COOT-FM2TEDL	91.80	15.19	3.90	86.52	10.86	3.30	92.52	15.27	3.91
SgO-SCNN	93.37	5.61	2.37	90.67	7.31	2.70	94.09	5.68	2.38
GuCA-KFDCN	93.98	6.25	2.50	91.34	7.93	2.82	94.82	6.26	2.50
ApR-FM2TEDL	94.81	6.57	2.56	92.19	8.24	2.87	95.57	6.85	2.62

5.14. Wilcoxon Tests

The Wilcoxon test is a paired test carried out to evaluate the statistical significance of ApR-FM2TEDL with other competing algorithms. In the Wilcoxon test, the performance of the outcomes attained from comparing the proposed ApR-FM2TEDL against baseline algorithms is validated, and the outcomes are shown in Table 6. Accordingly, the Probability values (p-values) of the proposed method showing the p-value indicates a difference of less than 0.05 between the number is considered as a statistically significant difference. Further, the Wilcoxon test in Table 7 demonstrates the significance of the proposed approach over the other baseline methods utilized for the comparison.

Table 7. Analysis based on the Wilcoxon tests

Methods/ Metrics	**Accuracy**		**Sensitivity**		**Specificity**	
	Wilcoxon-statistic	P-value	Wilcoxon-statistic	P-value	Wilcoxon-statistic	P-value
SVM-IDS	3.063	0.0011	2.738	0.0135	3.060	0.0011
RNN	3.061	0.0011	3.150	0.0014	3.060	0.0011
DNN	3.061	0.0011	3.150	0.0014	3.059	0.0012
BiLSTM-XAI	3.062	0.0011	3.150	0.0014	3.062	0.0011
KFDCN	3.089	0.0003	3.166	0.0019	3.089	0.0003
DCNN-BiLSTM-RNN	3.037	0.0019	3.035	0.0019	3.037	0.0019
SOA-KFDCN	2.957	0.0046	2.822	0.0098	2.956	0.0046
GEO-KFDCN	3.037	0.0019	3.035	0.0019	3.037	0.0019
OCSA-RNN	3.033	0.0020	2.944	0.0051	3.010	0.0028
S-SCAE+ Bi-DLDA	3.117	0.0005	3.017	0.0025	3.049	0.0015
ABC-FM2TEDL	3.002	0.0030	2.942	0.0051	3.001	0.0031
COOT-FM2TEDL	2.898	0.0068	2.669	0.0168	2.898	0.0068
SgO-SCNN	2.528	0.0246	1.914	0.0081	2.528	0.0246
GuCA-KFDCN	2.584	0.0214	1.987	0.0071	2.584	0.0214
ApR-FM2TEDL	2.607	0.0201	2.042	0.0065	2.589	0.0210

5.15. Friedman Tests

Statistical significance testing in terms of the Friedman test is conducted to check whether the difference in the outcomes between the proposed technique and other competing algorithms is significant or not. Specifically, the Friedman test provides the pair-wise comparison and ranking mechanism of the different attack detection algorithms. From the overall Analysis provided in Table 8, the p-values much lower than 0.05 are achieved by the proposed method, and hence the significant differences of the proposed method are explicated over the other competing algorithms.

Table 8. Analysis based on Friedman tests

Methods	**TP 80**					
	Accuracy		**Sensitivity**		**Specificity**	
	Friedman-statistic	P-value	Friedman-statistic	P-value	Friedman-statistic	P-value
SVM-IDS	3.226	0.0211	2.884	0.0335	3.223	0.0211
RNN	3.224	0.0211	3.318	0.0185	3.223	0.0211
DNN	3.224	0.0211	3.318	0.0185	3.222	0.0212

BiLSTM-XAI	3.225	0.0211	3.318	0.0185	3.225	0.0211
KFDCN	3.253	0.0203	3.335	0.0180	3.253	0.0203
DCNN-BiLSTM-RNN	3.199	0.0219	3.197	0.0219	3.198	0.0219
SOA-KFDCN	3.114	0.0246	2.972	0.0298	3.113	0.0246
GEO-KFDCN	3.199	0.0219	3.197	0.0219	3.198	0.0219
OCSA-RNN	3.194	0.0220	3.101	0.0251	3.171	0.0228
S-SCAE+ Bi-DLDA	3.283	0.0194	3.177	0.0225	3.211	0.0215
ABC-FM2TEDL	3.162	0.0230	3.098	0.0251	3.161	0.0231
COOT-FM2TEDL	3.052	0.0268	2.811	0.0368	3.052	0.0268
SgO-SCNN	2.663	0.0446	2.016	0.0101	2.663	0.0446
GuCA-KFDCN	2.721	0.0414	2.093	0.0091	2.721	0.0414
ApR-FM2TEDL	2.746	0.0401	2.151	0.0085	2.727	0.0410

5.16. Post-hoc Tests

Post-hoc tests serve as an effective tool for analyzing the significant difference between the group comparisons to interpret the results obtained with the different methods. Specifically, post-hoc analysis assists in identifying the specific groups that significantly influence the outcomes. Moreover, the post-hoc tests shown in Table 9 involve examining the p-values for evaluating the statistical significance. In the post-hoc tests, a p-value indicating a difference of less than 0.05 is considered a statistically significant difference.

Table 9. Analysis based on Post-hoc Tests

Methods	**TP 80**		
	Accuracy	**Sensitivity**	**Specificity**
	P-value	P-value	P-value
SVM-IDS	0.0311	0.0435	0.0311
RNN	0.0311	0.0285	0.0311
DNN	0.0311	0.0285	0.0312
BiLSTM-XAI	0.0311	0.0285	0.0311
KFDCN	0.0303	0.0280	0.0303
DCNN-BiLSTM-RNN	0.0319	0.0319	0.0319
SOA-KFDCN	0.0346	0.0398	0.0346
GEO-KFDCN	0.0319	0.0319	0.0319
OCSA-RNN	0.0320	0.0351	0.0328
S-SCAE+Bi-DLDA	0.0294	0.0325	0.0315
ABC-FM2TEDL	0.0330	0.0351	0.0331
COOT-FM2TEDL	0.0368	0.0468	0.0368
SgO-SCNN	0.0054	0.0111	0.0054
GuCA-KFDCN	0.0051	0.0101	0.0051
ApR-FM2TEDL	0.0050	0.0095	0.0051

6. Conclusion

The research developed the ApR-FM2TEDL framework for detecting attacks as well as mitigating attacks in cloud systems. The ApR-FM2TEDL method overcomes the hurdles faced by the conventional approaches regarding lower accuracy, lack of preprocessing, misdetections, and biased outcomes. The parallel processing of the ensemble methods with the minimum-maximum weighted strategy facilitated the understanding of diverse attack patterns as well as improved adaptability. The generation of synthetic samples prevented the overfitting issues by balancing the distribution of data. The developed ApRMO algorithm tuned the hyperparameters of the ApR-FM2TEDL system and, in addition, assisted in the production of synthetic samples. The utilization of the BAIT mitigation model enables the elimination of the attacks identified by the ApR-FM2TEDL framework and updates the information in the model to prevent future attacks from similar malicious packets. Furthermore, data undergoes missing imputation using the KNN technique that handles the non-linear boundaries of data distribution as well as enhances the data standard. The ApR-FM2TEDL is compared against the prevailing methods that show accurate detection efficiency with accuracy measures of 97.74%, sensitivity of 96.20%, and specificity of 98.59% in the BOT-IoT dataset. Future research can be conducted on the development of advanced feature extraction techniques for further boosting the overall efficiency.

References

[1] Anchal Ahalawat et al., "Entropy based DDoS Detection and Mitigation in OpenFlow Enabled SDN," *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, Vellore, India, pp. 1-5, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Wa'ad H. Aljuaid, and Sultan S. Alshamrani, "A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments," *Applied Sciences*, vol. 14, no. 13, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] S. Balasubramaniam et al., "Optimization Enabled Deep Learning-based DDoS Attack Detection in Cloud Computing," *International Journal of Intelligent Systems*, vol. 1, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Zhenyue Long et al., "A Transformer-based Network Intrusion Detection approach for Cloud Security," *Journal of Cloud Computing*, vol. 13, pp. 1-11, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] Karan B. Virupakshar et al., "Distributed Denial of Service (DDoS) Attacks Detection System for OpenStack-Based Private Cloud," *Procedia Computer Science,* vol. 167, pp. 2297-2307, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] M. Nadeem, A. Arshad, S. Riaz, S.S. Band, and A. Mosavi, "Intercept the Cloud Network from Brute Force and DDoS Attacks Via Intrusion Detection and Prevention System," *IEEE Access,* vol. 9, pp.152300-152309, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Runchuan Li et al., "An Intelligent Heartbeat Classification System based on Attributable Features with AdaBoost+Random Forest Algorithm," *Journal of Healthcare Engineering*, vol. 2021, no. 1, pp. 1-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Amira Mahamat Abdallah et al., "Cloud Network Anomaly Detection using Machine and Deep Learning Techniques—Recent Research Advancements," *IEEE Access*, vol. 12, pp.56749-56773, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Mohamed Ouhssini et al., "DeepDefend: A Comprehensive Framework for DDoS Attack Detection and Prevention in Cloud Computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 36, no. 2, pp. 1-25, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Sandeep Kumar Chundru et al., "Efficient Machine Learning Approaches for Intrusion Identification of DDoS Attacks in Cloud Networks," *SSRN*, pp. 1-7, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] M. Di Mauro et al., "Supervised Feature Selection Techniques in Network Intrusion Detection: A Critical Review," *Engineering Applications of Artificial Intelligence*, vol. 101, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Himanshu Setia et al., "Securing the Road Ahead: Machine Learning-Driven DDoS Attack Detection in VANET Cloud Environments," *Cyber Security and Applications*, vol. 2, pp. 1-11, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mona Alduailij et al., "Machine-Learning-based DDoS Attack Detection using Mutual Information and Random Forest Feature Importance Method," *Symmetry*, vol. 14, no. 6, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] S.G. Kene, and D.P. Theng, "A Review on Intrusion Detection Techniques for Cloud Computing and Security Challenges," *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India, pp. 227-232, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[15] M. Mehmood, R. Amin, M.M.A. Muslam, J. Xie, and H. Aldabbas, "Privilege Escalation Attack Detection and Mitigation in Cloud Using Machine Learning," *IEEE Access*, vol. 11, pp. 46561-46576, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Kaneez Fizza et al., "QoE in IoT: A Vision, Survey and Future Directions," *Discover Internet of Things*, vol. 1, pp. 1-14, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Nishika Gulia et al., "Intrusion Detection System Using the G-ABC with Deep Neural Network in Cloud Environment," *Scientific Programming*, vol. 2023, no. 1, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Alaa Alsaeedi, Omaimah Bamasag, and Asmaa Munshi, "Real-Time DDoS Flood Attack Monitoring and Detection (RT-AMD) Model for Cloud Computing," *Proceedings of the 4th International Conference on Future Networks and Distributed Systems*, pp. 1-5, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Muhammad Sajid et al., "Enhancing Intrusion Detection: A Hybrid Machine and Deep Learning Approach," *Journal of Cloud Computing*, vol. 13, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Ramin Fadaei Fouladi, Orhan Ermiş, and Emin Anarim, "A DDoS Attack Detection and Defense Scheme using Time-Series Analysis for SDN," *Journal of Information Security and Applications*, vol. 54, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Can Zhao et al., "Secure Consensus of Multi-Agent Systems with Redundant Signal and Communication Interference via Distributed Dynamic Event-Triggered Control," *ISA Transactions*, vol. 112, pp. 89-98, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Hanaa Attou et al., "Cloud-based Intrusion Detection approach using Machine Learning Techniques," *Big Data Mining and Analytics*, vol. 6, no. 3, pp. 311-320, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Abdel-Rahman Al-Ghuwairi et al., "Intrusion Detection in Cloud Computing Based on Time Series Anomalies Utilizing Machine Learning," *Journal of Cloud Computing*, vol. 12, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Abdelrahman Elsharif Karrar, "The Effect of using Data Pre-Processing by Imputations in Handling Missing Values," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 10, no. 2, pp.375-384, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Nishit Patil, and Shubhlaxmi Joshi, "ChSp-ACN:: Channelized Spatial Attention Enabled Adam-Optimized Convolutional Neural Network for Intrusion Detection," *Cybernetics and Information Technologies*, vol. 26, no. 1, pp. 1-18, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[26] Tao Wu et al., "Intrusion Detection System Combined Enhanced Random Forest with SMOTE Algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[27] Dina Elreedy, Amir F. Atiya, and Firuz Kamalov, "A Theoretical Distribution Analysis of Synthetic Minority Oversampling Technique (SMOTE) for Imbalanced Learning," *Machine Learning*, vol. 113, pp. 4903-4923, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jiyeon Kim et al., "CNN-Based Network Intrusion Detection Against Denial-of-Service Attacks," *Electronics*, vol. 9, no. 6, pp. 1-21, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[29] Yiying Zhang et al., "A BiLSTM-based DDoS Attack Detection Method for Edge Computing," *Energies*, vol. 15, no. 21, pp.1-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[30] Alaa Q. Raheema, "Improved Security in Cloud Computer Networks Using RNN Deep Learning Techniques," *Journal of Cybersecurity & Information Management*, vol. 15, no. 1, pp. 365-384, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[31] Abdulbari Talib Naser et al., "Improved Coot Optimizer Algorithm-based MPPT for PV Systems under Complex Partial Shading Conditions and Load Variation," *Energy Conversion and Management: X*, vol. 22, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[32] Eric Forcael et al., "Artificial Bee Colony Algorithm to Optimize the Safety Distance of Workers in Construction Projects," *Mathematics*, vol. 12, no. 13, pp. 1-23, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[33] DDoS evaluation dataset (CIC-DDoS2019), University of New Brunswick. [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html

[34] The Bot-IoT Dataset, UNSW Sydney, [Online]. Available: https://research.unsw.edu.au/projects/bot-iot-dataset

[35] Ayoub Alsarhan et al., "Machine Learning-Driven Optimization for SVM-based Intrusion Detection System in Vehicular Ad Hoc Networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 6113-6122, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[36] Sharuka Promodya Thirimanne et al., "Deep Neural Network based Real-Time Intrusion Detection System," *SN Computer Science*, vol. 3, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[37] S. Sivamohan, and S.S. Sridhar, "An Optimized Model for Network Intrusion Detection Systems in Industry 4.0 using XAI based Bi-LSTM Framework," *Neural Computing and Applications*, vol. 35, pp. 11459-11475, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[38] A. Karthick Kumar et al., "Enhanced Hybrid Deep Learning Approach for Botnet Attacks Detection in IoT Environment," *2024 7th International Conference on Signal Processing and Information Security (ICSPIS)*, Dubai, United Arab Emirates, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[39] Gaurav Dhiman, and Vijay Kumar, "Seagull Optimization Algorithm: Theory and its Applications for Large-Scale Industrial Engineering Problems," *Knowledge-based Systems*, vol. 165, pp. 169-196, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[40] Reddy SaiSindhuTheja, and Gopal K. Shyam, "An Efficient Metaheuristic Algorithm based Feature Selection and Recurrent Neural Network for DoS Attack Detection in Cloud Computing Environment," *Applied Soft Computing*, vol. 10, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[41] Andrea Sharon et al., "An Intelligent Intrusion Detection System using Hybrid Deep Learning Approaches in Cloud Environment," *International Conference on Computer, Communication, and Signal Processing*, pp. 281-298, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[42] Abdolkarim Mohammadi-Balani et al., "Golden Eagle Optimizer: A Nature-Inspired Metaheuristic Algorithm," *Computers & Industrial Engineering*, vol. 152, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[43] Yogesh B. Sanap, and Pushpalata G. Aher "Seagull-optimized Deep Convolutional Neural Network for Detecting Distributed Denial of Service Attack in Cloud Computing," *2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, Erode, India, pp. 35-42, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[44] Moinul Islam Sayed et al., "A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network," *2022 International Wireless Communications and Mobile Computing (IWCMC)*, Dubrovnik, Croatia, pp. 1125-1130, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[45] Md. Alamgir Hossain, and Md. Saiful Islam, "Enhancing DDoS Attack Detection with Hybrid Feature Selection and Ensemble-Based Classifier: A Promising Solution for Robust Cybersecurity," *Measurement: Sensors*, vol. 32, pp. 1-12, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[46] Ahmed Ahmim et al., "Distributed Denial of Service Attack Detection for the Internet of Things using Hybrid Deep Learning Model," *IEEE Access*, vol. 11, pp. 119862-119875, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[47] Chin-Shiuh Shieh et al., "Detection of Unknown DDoS Attacks with Deep Learning and Gaussian Mixture Model," *Applied Sciences*, vol. 11, no. 11, pp. 1-13, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[48] Marcos V.O. de Assis et al., "Near Real-Time Security System Applied to SDN Environments in IoT Networks Using Convolutional Neural Network," *Computers & Electrical Engineering*, vol. 86, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[49] Hassan A. Alamri, and Vijey Thayananthan, "Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-Defined Networks Against DDoS Attacks," *IEEE Access*, vol. 8, pp. 194269-194288, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[50] S. Velliangiri et al., "Detection of Distributed Denial of Service Attack in Cloud Computing using the Optimization-Based Deep Networks," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 33, no. 3, pp. 405-424, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[51] E. Arul, and A. Punidha, "Supervised Deep Learning Vector Quantization to Detect MemCached DDOS Malware Attack on Cloud," *SN Computer Science*, vol. 2, pp. 85-12, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[52] Muhammad Zeeshan et al., "Protocol-Based Deep Intrusion Detection for DoS and DDoS Attacks Using UNSW-NB15 and Bot-IoT Data-Sets," *IEEE Access*, vol. 10, pp. 2269-2283, 2022. [CrossRef] [Google Scholar] [Publisher Link]