



Modelling an Improved Swarm Optimizer and Boosted Quantile Estimator For Malicious Flow Monitoring And Prediction In Network

U. Harita, Moulana Mohammed*

Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, India

Emails: uharita@gmail.com; moulanaaphd@gmail.com

Abstract

For a long time, malware has posed a significant risk to computer system security. The effectiveness of conventional detection techniques based on static and dynamic analysis is restricted due to the quick advancement of anti-detection technologies. In recent years, AI-based malware detection has increasingly been employed to combat malware due to its improved predictive ability. Unfortunately, because malware may be so diverse, it can be challenging to extract features from it, which makes using AI for malware detection ineffective. A malware classifier based on an Improved Salp Swarm optimization for feature selection and a Boosted tree with Conditional Quantile Estimation (ISSO-BCQE) is developed to adapt different malware properties to solve the problem. Specifically, the malware code is extracted, and the feature sequence is generated into a boosting tree where the feature map of the node is extracted using BCQE, where a boosting network is used to design a classifier and the method's performance is finally analyzed and compared. The results show that our model works better than other approaches regarding FPR and accuracy. It also shows that the method beats current methods with the highest accuracy of 99.6% in most detecting circumstances. It is also stable in handling malware growth and evolution.

Keywords: IoT; network traffic; feature selection; classification; feature optimization

1. Introduction

Researchers have been trying to address the issue of information security through the use of more advanced technology, and they have produced several research findings that effectively reduce the harm caused by malevolent network attacks, even though information security departments spend a substantial amount of money annually maintaining network security [1]. However, due to the distinct characteristics of information technology, malicious programmers frequently search aggressively for system weaknesses to find a breakthrough. However, after the system is compromised and users are lost, the security staff often look at ways to stop and neutralize malware [2]. As such, the security mechanisms in place today are frequently designed with "hindsight" and passive defence in mind. Information security research has begun to focus heavily on anticipating the characteristics of malware, identifying it early, and stopping it in its tracks [3].

The majority of defensive systems continue to rely mostly on static analysis, which for their measures is based on signature matching [4]. Sensitive actions, gaining access to significant rights, network analysis, and crucial process monitoring are all examples of dynamic analysis created by some defensive systems as a helping technique [5]. To be restricted in their ability to defend against new malware kinds or variations, these techniques are primarily targeted at certain malware or malware classes. Additionally, they are vulnerable to

anti-detection strategies, which may allow malware to masquerade as something else and trick the detectors into harming [6]. These scenarios all point to the necessity of creating a novel detection technique. Deep learning technology has demonstrated previously unheard-of levels of intelligence in recent years [7]. Examining extensive past data can identify patterns and forecast unidentified samples. It has high levels of intelligence, predictive ability, and adaptability and has demonstrated achievement in several domains, such as natural language processing, computer vision, and speech recognition [8].

Furthermore, they are susceptible to anti-detection techniques, which might enable malware to assume a different identity and fool the detectors into inflicting harm [9]. These scenarios all point to the necessity of creating a novel detection technique. Current times have seen unprecedented levels of intelligence displayed by deep learning technologies. It can anticipate unknown samples and find trends by looking through a large amount of historical data [10]. It has proven effective in several fields, including natural language processing, computer vision, and speech recognition. It possesses high degrees of intelligence, flexibility, and predictive capacity. We introduce a novel approach for feature extraction from three-dimensional structured data [11] – [13]. Using the API call records of several known malware samples, we first extract the samples' weight models using a Markov chain. Then, to remove the properties of the samples that need to be detected, we use the samples that need to be recognized to map into the weight model [14]. The model combines general malware traits with test sample features to create freshly produced features that effectively withstand virus variations and disguise themselves while retaining the qualities of the sample itself [15]. A technique for detecting malicious code that utilizes ISSO-BCQE is suggested. The BCQE-based malware detection model is developed by training and testing the boosting network using the characteristic harmful leaf node as the input. To increase the model's versatility in identifying malicious code, the model uses the boosted tree as the ISSO-BCQE input.

The work is modelled as follows: section 2 provides a broader analysis of various prevailing approaches. The methodology is drafted in section 3, and experimental outcomes in section 4. The work summary is provided in section 5.

2. Related works

The growing user base and volume of traffic created by communication systems and networks pose several ongoing challenges for NTMA, comprising: (1) collecting and archiving traffic data; (2) using traffic data to achieve corporate goals by obtaining insights; (3) incorporating traffic data; (4) verifying traffic data; (5) safeguarding traffic data; and (6) purchasing traffic data [16]. There is a need for ongoing research to evaluate and track networking performance because of the extraordinary rise in the volume of data and the number of linked nodes [17]. Furthermore, the accessibility of enormous and varied volumes of traffic data calls for adopting novel techniques for network management data monitoring and analysis. Owing to these difficulties, most efforts concentrate on a solitary NTMA component, such as quality of service, anomaly detection, or traffic classification [18] – [19].

Some previously listed obstacles are traffic data collecting poses significant technological hurdles in network traffic management analytics (NTMA). This is especially true for active measurements, where probes must be used to monitor the evolution of critical network parameters over time. One of the best ways to get information about the end-users overall experience with performance is through probes. Active and passive probing are two popular techniques that deliver precise traffic data and may increase the effectiveness of end-to-end measurement to identify QoS and QoE [20]. An active probe simulates and distributes network traffic to evaluate end-to-end performance (such as latency). Passive probes offer an alternative perspective on the network to active probes. Positioned across network connections, passive probes enquire about traffic passing via the connection under observation. For network traffic collecting, network packets remain the most used data format. Nonetheless, packet loss is an issue that most network packet-collecting techniques face, especially when dealing with high traffic volumes [21]. Furthermore, these techniques become unusable owing to their limited competence and struggle with high-speed connectivity [22]. Another well-liked method for acquiring data is through flow-based strategies. A collection of network packets with identical source and destination IP addresses and ports is known as a flow network [23]. Flow-based data-gathering approaches can perform better than packet-based procedures and reduce the required packet analysis procedures, especially in Gigabit Networks. Nevertheless, flow and packet filtering can significantly threaten these methods. Numerous survey studies have been published on methodologies, structures, and strategies for acquiring traffic statistics [24] – [25].

The research community has given deep learning much attention because of its growing popularity. Due to its ability to automatically classify network traffic (whether conventional or network traffic) [26], data about internet traffic may also be processed using deep learning. Furthermore, the increasing accessibility and capability of the Graphics Processing Unit (GPU) contribute significantly to the acceleration of large-scale mathematics and matrix computations, hence offering a substantial amount of resources that align with the needs of deep learning-based techniques [27]. Based on the underlying model, deep learning for network traffic detection is typically divided into many categories, including unsupervised learning, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) [28]. Very few papers are published for deep learning-based deep web identification, according to a review of earlier literature. However, a new deep learning model may be constructed and tested offline by utilizing the flow-based data—gathering packets of various flows over time and then categorizing them into distinct flow packets. The system needs more time to collect traffic flow statistics if a big session is related to the flow. Furthermore, extracting features from the flow-based traffic data takes time. In summary, the entire profiling process to detect network traffic is time-consuming. Again, high precision in resources is required to categorize either network or organic traffic. These resources include large amounts of memory, powerful computers, the ability to receive and handle accumulated traffic, etc. Numerous studies have demonstrated that watching a small number of data packets is sufficient to assess the problem, as opposed to seeing every packet in a flow to compare it to traditional or network traffic [29] – [30].

Thus, there is a need for a new system that can analyze internet traffic streams and identify and extract patterns from them to address the problems above. As such, we provide a unique transfer learning methodology in this study that enables identifying and forecasting malicious traffic inputs from the network. A portion of authorized IP and route space that is empty of servers and live services is known as a network. The network does not respond to queries or assist sections of the overlay network in gaining access to non-standard communication protocols and ports, even though it has a concealed mechanism for receiving messages. Network network activity is typically malevolent and suspicious. Systems can be built to monitor networks to draw in intelligence spies or apprehend attackers. Malware and botnets are often less clever than humans and are responsible for propagating internet traffic, including network traffic.

3. Methodology

3.1. Dataset

The assessment and comparison of the various models use one of the datasets most often utilized in intrusion detection systems, the CICIDS2017 dataset. The dataset is created by collecting network data over four days, each day focusing on a different form of attack or typical activity. Table 1 and Table 2 shows how the CICIDS2017 dataset's data is distributed. It contains a lot of data, but some need to be labelled, damaged, or duplicated, and data distribution across the classes must be more balanced. A cleaned version of the dataset was utilized to address the problem of duplicate and unlabeled records. There are 78 features in the updated version.

3.2. Salp swarm optimization for choosing features

SSA was the name of one of the random population-based algorithms. SSA mimics how salps congregate in the water to hunt their food. Salps usually form a salp chain, a swarm in heavy seas. The salps that trail behind are called followers. According to the SSA algorithm, the leader is the salp at the top of the chain. Salps's position is inside an s –dimensional search space, where s is the size of the problem variable count, as in earlier swarm-based techniques. Thus, a two-dimensional matrix named z contains the position of every salp. Additionally, it is presumable that the swarm aims for a food supply in the P search area. The SSA mathematical model is provided as follows:

$$z_n^1 = \begin{cases} P_n + r_1((u_n - l_n)r_2 + l_n) & r_3 \geq 0 \\ P_n - r_1((u_n - l_n)r_2 + l_n) & r_3 < 0 \end{cases} \quad (1)$$

The following equation, where all symbol meanings are displayed in Table 1, can be used by the leader salp to shift positions.

$$r_1 = 2e \left(-\frac{4a}{A} \right)^2 \quad (2)$$

Because it balances the capacities for exploration and extraction, the coefficient r_1 is the most important parameter in SSA. The following equations are applied to modify the followers' position:

$$z_n^m = \frac{1}{2}ce^2 + v_0e \quad (3)$$

Where, $m \geq 2, c = \frac{v_{final}}{v_0}$ in which $v = \frac{z-z_0}{e}$. Since optimization involves iterations, assuming $v_0 = 0$ is taken into account, the equation expressing the contradiction between them is equivalent to 1:

$$z_n^n = \frac{1}{2}(z_n^m + z_n^{m-1}) \quad (4)$$

This method may be described in depth, step-by-step, as follows.

1. Set up the SSA's parameters, including the best fitness value ($f(Z^*)$), optimal salp position (Z^*), the number of salps (S) and iterations (A).
2. Begin a population of S salp's places at random.
3. Consider each salp's fitness.
4. Define the iteration count (a) to zero.
5. Apply Eq. (2) to $r1$.
6. With every salp,
 - (a) Use Eq. (1) to update the leading salp's location if $m == 1$.
 - (b) If not, use Eq. (4) to update the follower salp's location.
 - (c) Assess the suitability of each salp.
 - (d) If a better option turns up, update Z^* .
7. Raise by a single.
8. Repeat steps five through seven until $a = A$ is reached.
9. Give back the fitness value $f(Z^*)$ and the optimal solution Z^* .

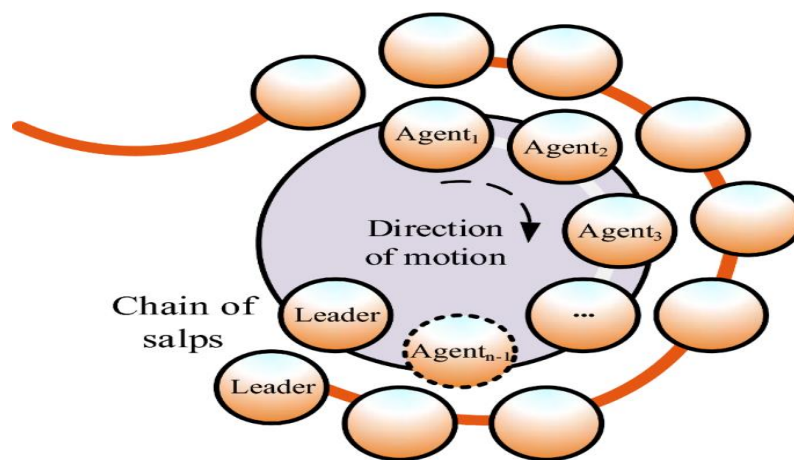


Figure 1: Improved salp swarm optimization

Table 1: Parameters

Parameters	Value	Meaning
ρ	0.9	Constant
$N - \text{iterations}$	50	Maximal iterations
$N - \text{agents}$	10	Population size
$N - \text{runs}$	20	No. of runs
Dimension		No. of features
Searching		Vector [0,1]
K	5	k-value
ω	0.7	ISSO
$c1, c2$	1.4	Learning factors
Cross_val	0.9	Cross-over
Mut_val	0.1	mutation

3.3. Convergence analysis

Salps' location is updated by Eq. (1) and (4). The most optimal solution at the moment determines the updated solution in SSA. Similar to the PSO method, one obtains the improved salp swarm method (ISSA) by adding an inertia weight $x_2[0,1]$ into the SSA. Increase the convergence rate when searching is achieved via the ISSA algorithm's new parameter. To precisely identify the ideal answer and avoid the abundance of local solutions that result from feature selection tasks, it also finds a balance between the capabilities of exploration and exploitation. These equations represent the enhanced algorithm:

$$Z_n^1 = \begin{cases} \omega P_n + r_1 ((u_n - l_n)r_2 + l_n) & r_3 \geq 0 \\ \omega P_n - r_1 ((u_n - l_n)r_2 + l_n) & r_3 < 0 \end{cases} \quad (5)$$

$$z_n^m = \frac{1}{2} (z_n^m + \omega Z_n^{m-1}) \quad (6)$$

3.4. Strengthened tree using Conditional Quantile Approximation

In this part, the *gbex* technique estimates the GPD parameters $(\sigma(x), \gamma(x))$ and constructs a gradient-boosting group of tree forecasters. The method is the traditional Friedman boosting approach, defined by Friedman using the GPD negative log-likelihood, also referred to as deviation in the machine learning community serving as the aim function. Two series of trees are required for the GPD since it has two parameters; this is comparable to the multiclass classification technique, which trains several tree sequences to understand the various class probabilities. This work uses a (non-extreme) quantile regression approach to estimate the intermediate quantile function $x \mapsto \hat{Q}_x(\tau_0)$ and compute the exceedance sample. We may obtain the exceedances above the intermediate threshold shown by using this estimated function on the predictor variables $X_1, \dots, X_n$.

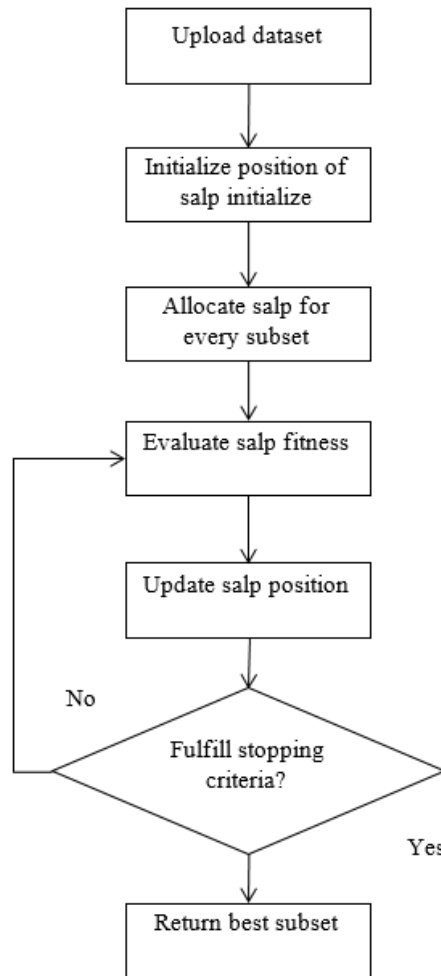


Figure 2: Flow diagram of ISSO

$$Z_i = \left(Y_i - \hat{Q}_x(\tau_0) \right)_+, i = 1, \dots, n \quad (7)$$

Suppose Y_i falls below the barrier, $Z_i = 0$. We take for granted that the intermediate threshold is large enough to let the generalized Pareto distribution explain the exceedances and that the approximation of conditional quantiles is accurate. Our objective is to obtain the conditional parameter $\theta(x) = (\sigma(x), \gamma(x))$ using the sample of exceedances over the threshold. We aim to lower the negative log-likelihood or GPD deviation using Friedman's tree-base gradient boosting technique. The maximum likelihood approach is a typical technique for estimating the GPD parameters $\theta = (\sigma, \gamma)$ without covariates. This method yields asymptotically standard estimators in the unconditional situation where $\gamma > -1/2$. With parameters $\theta(X_i) = (\sigma(X_i), \gamma(X_i))$, the negative log-likelihood or deviation for exceedances from GPD distribution provided by:

$$\ell_{Z_i}(\theta(X_i)) = \left[(1 + 1/\gamma(X_i)) \log \left(1 + \gamma(X_i) \frac{Z_i}{\sigma(X_i)} \right) + \log \sigma(X_i) \right] 1_{Z_i > 0} \quad (8)$$

The greatest likelihood estimator without conditions provides the initial estimate used by the gradient boosting technique, which is:

$$\theta_0(x) = \theta_0 = \arg \min_{\theta} \sum_{i=1}^n \ell_{Z_i}(\theta) \quad (9)$$

Then, an incremental improvement is made to this originally constant model over time. This work gradually construct a series of B pairs of trees, starting with the initial constant model $\theta_0 = (\sigma_0, \gamma_0)$. The objective of step b is to add two gradient trees $(T_b^\sigma(x), T_b^\gamma(x))$ to the existing model $\theta_{b-1}(x) = (\sigma_{b-1}(x), \gamma_{b-1}(x))$ to enhance it. If $b = 1, \dots, B$:

- i) From the collection of exceedances $(X_i, Z_i)_{1 \leq i \leq n}$, a subsample is chosen at random;
- ii) The residuals or derivatives of deviations about σ and γ are computed on this subsample;
- iii) Two distinct leaf partitions of the feature space are obtained by fitting two regression trees to the residuals;
- iv) The model has a reduced representation of both trees. The line search approximation is used to modify the tree values on each leaf to minimize the deviation.

Now, the model gives specific mathematical information for each stage. The traditional regression tree relies on square loss minimization. While $b = 1, \dots, B$ then

- i) A random subset $S_b \subset \{1, \dots, n\}$ of size $[sn]$ is chosen at random with $s \in [0, 1]$ serving as the subsampling fraction parameter.

- ii) On the sub-sample S_b , the model residuals are calculated by:

$$r_{b,i}^\sigma = \frac{\partial \ell_{Z_i}}{\partial \sigma} (\theta_{b-1}(X_i)) \quad (10)$$

$$r_{b,i}^\sigma = \frac{\partial \ell_{Z_i}}{\partial \sigma} (\theta_{b-1}(X_i)); \quad i \in S_b \quad (11)$$

- iii) On $(X_i, r_{b,i}^\sigma)_{i \in S_b}$ and $(X_i, r_{b,i}^\gamma)_{i \in S_b}$ are two regression trees $(T_b^\sigma(x), T_b^\gamma(x))$ are fitted. The tree's construction, which divides the feature area into several leaves, uses the traditional CART technique. It uses recursive binary splitting as its foundation. The halting rule depends on several factors encompassing the L_{min}^σ min and L_{min}^γ minimal leaf sizes (minimum number of observations per leaf) and the greatest depths D_p, D_σ (the maximum number of splits that occur in a tree between a root and a leaf). $(L_{b,j}^\sigma)_{1 \leq j \leq J_b^\sigma}$ ($(L_{b,j}^\gamma)_{1 \leq j \leq J_b^\gamma}$) represents the leaves of T_b^σ (resp. T_b^γ). The regression trees are then altered: the partitions remain unaltered; however, the tree values are chosen to reduce the divergence. The process for doing this is line search, whereby the minimization issue is solved to yield the updated value, $\xi_{b,j}^\sigma$ in $L_{b,j}^\sigma$.

$$\xi_{b,j}^\sigma = \arg \min_{\xi} \sum_{X_i \in L_{b,j}^\sigma} \ell_{Z_i}(\theta_{b-1}(X_i) + \xi_{\ell_\sigma}), \quad j = 1, \dots, J_b^\sigma \quad (12)$$

Where the line search's direction corresponds to σ is indicated by the symbol $\ell_\sigma = (1, 0)$. The value $L_{b,j}^\gamma$ is obtained in the leaf $\xi_{b,j}^\gamma$ (everywhere σ is replaced for ξ) as a result of the line search for the parameter σ , which is conducted in the direction $e_\gamma = (0, 1)$. The line search could be computationally costly; hence, an approximation—a Newton-Raphson step is utilized.

$$\xi_{b,j}^\sigma = - \frac{\sum_{X_i \in L_{b,j}^\sigma} \frac{\partial \ell_{Z_i}}{\partial \sigma} (\theta_{b-1}(X_i))}{\sum_{X_i \in L_{b,j}^\sigma} \frac{\partial^2 \ell_{Z_i}}{\partial \sigma^2} (\theta_{b-1}(X_i))} \quad (13)$$

The value $\xi_{b,j}^\gamma$ (the identical equation with σ substituted for γ everywhere) is obtained from the leaf's estimate for line searches $L_{b,j}^\gamma$ for the parameter γ . The gradient trees may be found using Eq. (14):

$$T_b^\sigma = \sum_{j=1}^{J_b^\sigma} \tilde{\xi}_{b,j}^\sigma \mathbb{I}_{\{x \in L_{b,j}^\sigma\}} \quad (14)$$

$$T_b^\gamma(x) = \sum_{j=1}^{J_b^\gamma} \tilde{\xi}_{b,j}^\gamma \mathbb{I}_{\{x \in L_{b,j}^\gamma\}} \quad (15)$$

iv) Finally, the model $\theta_{b-1}(x) = (\sigma_{b-1}(x), \gamma_{b-1}(x))$ is revised by:

$$\theta_b(x) = (\sigma_b(x), \gamma_b(x)) \quad (16)$$

Where learning rates are defined as the shrinkage parameters, $\lambda^\sigma, \lambda^\gamma \in (0,1)$. Since the present model adds a reduced version of the trees, they are utilized to slow down the dynamics. The gradient boosting model is the result of the calculated parameters.

$$\hat{\sigma}(x) = \sigma_0 + \lambda^\sigma \sum_{b=1}^B T_b^\sigma(x) \quad (17)$$

$$\hat{\gamma}(x) = \gamma_0 + \lambda^\gamma \sum_{b=1}^B T_b^\gamma(x) \quad (18)$$

The gradient boosting approach has been devised to fit the conditional GPD model utilizing the previously described method for exceedances exceeding a threshold. The gradient of the GPD negative log-likelihood may inflate, and it is not Lipschitz, which might lead to numerical instability in the initial implementation of this technique. Consequently, we provide gradient clipping, a standard machine-learning method to prevent gradient explosion. To lessen the significant impact of extreme observations, As a result, we limit the Newton-Raphson step's absolute value to 1.

$$T_b^\sigma(x) = \sum_{j=1}^{J_b^\sigma} \text{sign}(\tilde{\xi}_{b,j}^\sigma) \min(|\tilde{\xi}_{b,j}^\sigma|, 1) \mathbb{I}_{\{x \in L_{b,j}^\sigma\}} \quad (19)$$

Similarly, for $T_b^\gamma(x)$. In actual use, this work finds that using gradient clipping yields a more reliable and effective method. Algorithm 1 provides a summary of the GPD exceedance modelling process. The quantity of B iterations is a crucial metric in real-world applications, and choosing it requires striking a balance between variance and bias, as $B \rightarrow \infty$, the method is susceptible to over-fitting. To avoid this, early termination via cross-validation is achieved; here, we address the pragmatic interpretation and choice of the various tuning parameters.

Algorithm 1:

Input: Define initial parameter values, fix threshold value, no. of gradient tree, maximal tree depth, learning rate, minimal leaf size (nodes)

1. **For** $b = 1, \dots, B$
2. Perform random sub-sample $S_b \subseteq \{1, \dots, n\}$ of size $[sn]$
3. Evaluate deviance based on provided samples;
4. Execute regression tree;
5. Predict gradients using covariates function;
6. Evaluate maximal tree depth, minimal leaf nodes;
7. Approximate gradient function;
8. Update GPD parameters based on learning rates;

Output: Conditional parameters; //GPD

9. **End for**

4. Numerical results and discussion

The categorization of network traffic in our experiment was done using publicly available data. From the first network data, eight traffic groups were selected. The collection comprises VPN and TOR programs that simulate network traffic, compiled from both public datasets CICIDS2017, like items from the datasets that are malicious and benign. Voice over Internet Protocol (VOIP), Chat, Email, P2P, Video-Stream, Audio-Stream, and Browsing are all malicious network activity. The dataset's specific attributes are displayed in Table 1. Time-related characteristics, computed by altering the time, are included in each dataset instance, such as the duration of a flow's activity, the intervals between packets, etc. Furthermore, fixed time-based network parameters are considered, such as bytes or packets per second. A few earlier researches provide more details concerning data. Heat maps are a potential tool for displaying a broad overview of data distribution since they are self-explanatory. They use colour variations to illustrate data. Here, the network's traffic occurrences' intensity is shown using heat maps. Blue hues represented the traffic occurrences. Understanding the data distribution through visualization can be aided by this. Ten distinct pre-trained networks are used in this investigation; Table 1 lists these networks together with the needed input size, number of layers, and network depth, and each pre-trained network has a different set of final layers. Improved salp models produce optimization methods based on the target function's probability model to provide better options for assessing the subsequent hyperparameter set.

To find out how well various pre-trained transfer learning models performed using the baseline classifier, the baseline classifiers were tested using a 10-fold validation model. 90% of the data was used to train all classifiers in the transfer learning framework, and 10% of the data was utilized for testing, followed by ten folds. The baseline methods' different parameter values are displayed in Table 1. Interestingly, the ISSO optimizer correctly adjusted the comparing algorithms' parameters to get the optimum classification rate in our experimental experiments. The most recent advanced parameters were established based on the algorithms under comparison to provide an equitable comparison. Among the measures were the following: F-score, sensitivity (SN), specificity (SP), negative predictive value (NPV), positive predictive value (PPV), overall prediction accuracy (ACC). Some metrics are used to assess the effectiveness of the suggested approach. This work also used 10-fold cross-validation. These parameters are defined below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (20)$$

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

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

$$F1 - measure = \frac{2TP}{2TP + FP + FN} \quad (23)$$

$$Precision = \frac{TP}{TP + FP} \quad (24)$$

Here, TP and TN represent the proportion of harmful and non-malicious traffic successfully anticipated, whereas FP and FN represent the traffic forecasted wrongly. The data format following data conversion was examined, and the single instance's structure was shown using the bar plots in Fig 6 to Fig 8. The model produced by the proposed approach is demonstrated in Fig. 6. The data transformations for the instance are shown in Fig 5. The samples are mapped for each dataset instance, which, in turn, correspond to existing approaches. The one-dimensional dataset samples are provided in the direction (x-axis) along which the dataset is non-linearly separable; the direction orthogonal to the x-axis along which the dataset displays the most significant variance is the second dimension (y-axis). Given probability distributions in the upper two dimensions of the x- and y-axes, the existing approach decreases the dimensions by doing so.

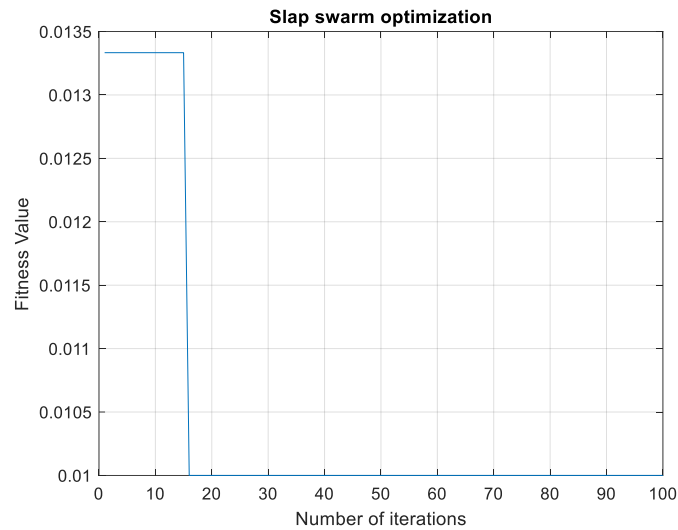


Figure 3: No. of iterations vs. Fitness value computation for feature selection

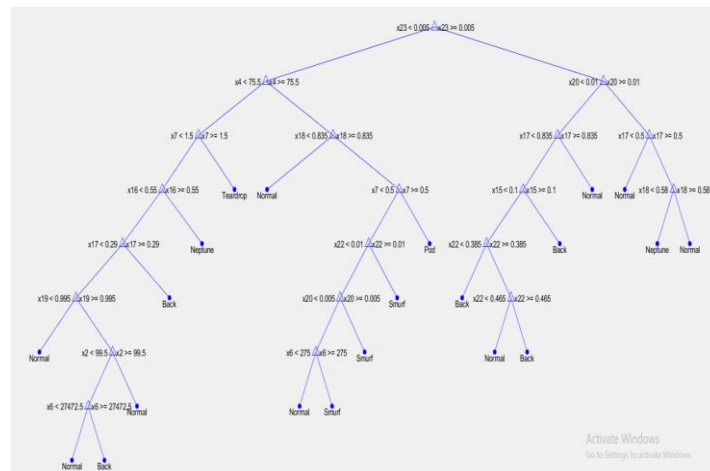


Figure 4: Tree based Conditional Quantile Approximation

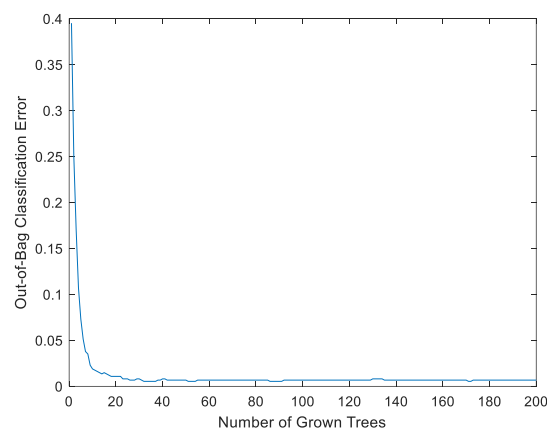


Figure 5: No. of grown trees vs. out-of-bag classification error

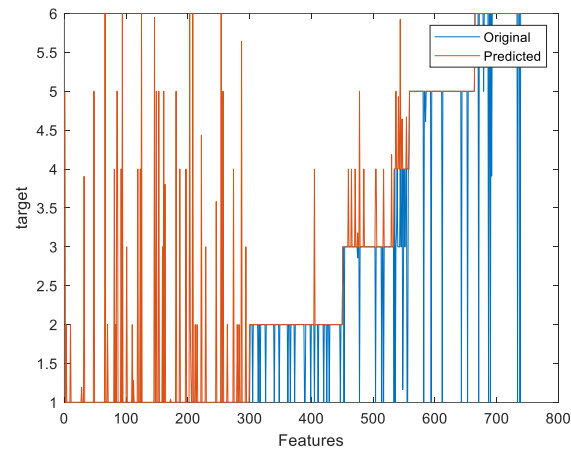


Figure 6: Original vs. predicted features

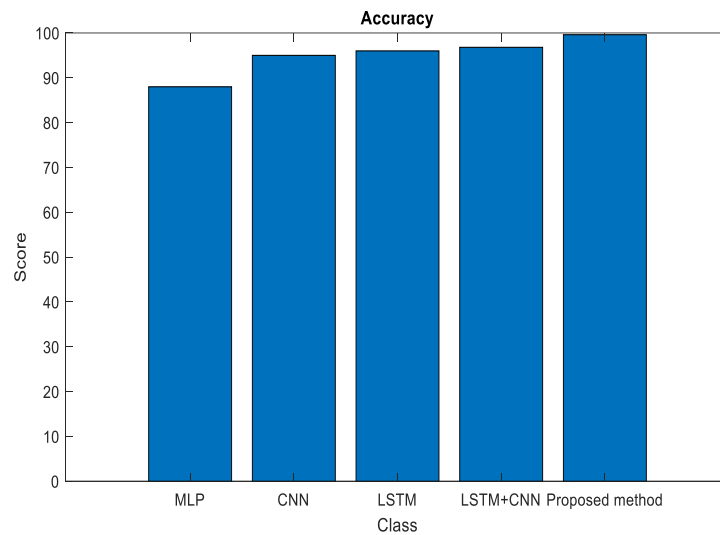


Figure 7: Accuracy comparison

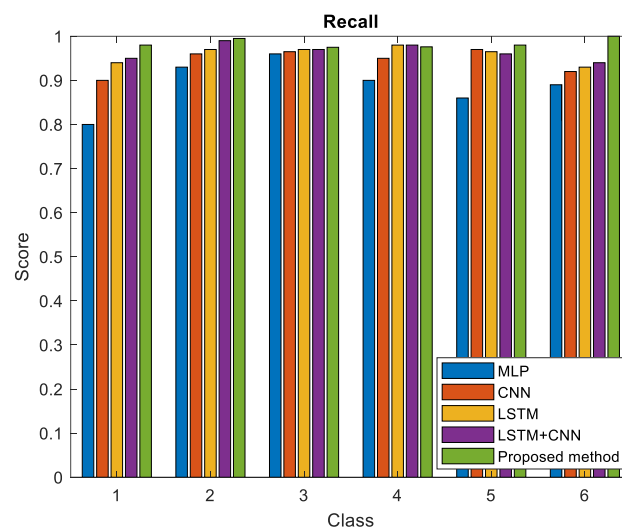


Figure 8: Recall comparison

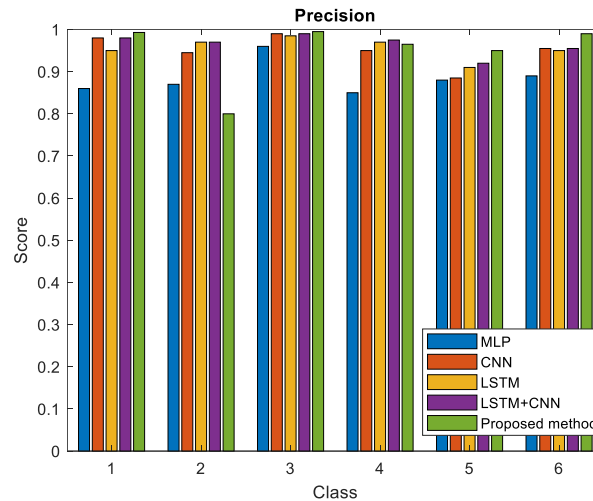


Figure 9: Precision comparison

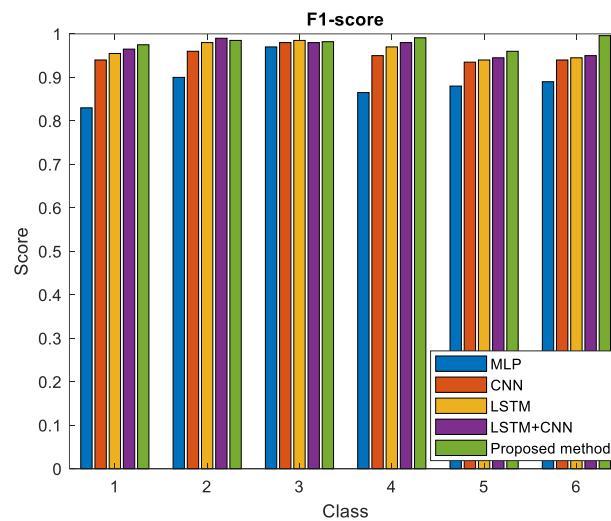


Figure 10: F1-score comparison

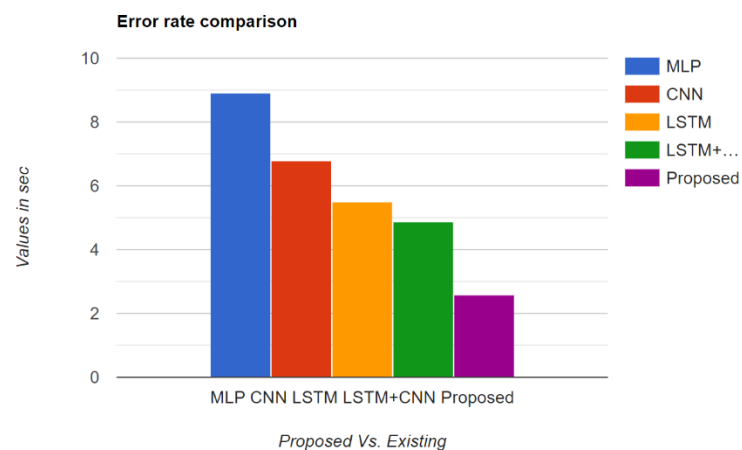


Figure 11: Error rate comparison

4.1. Discussion

This work thoroughly assessed the suggested categorization model for learning and optimization using the standard network dataset to predict harmful activities. Fig 7 to Fig 11 displays the proposed learning paradigm's expected effectiveness. Similar measurements are made for. Accuracy is calculated as the mean of ten runs of 10-fold validation accuracy. The results of parameter adjustment for the ISSO optimizer are displayed in Fig 3. Every baseline classifier underwent tuning to find the best parameters with a maximum iteration limit of thirty. A box plot is also used in Fig 5 to illustrate a traffic categorization performance metric. Lastly, the provided figures display the performance comparison using the most advanced methods on the time-based characteristics; six baseline classifiers are used. The following points might be emphasized in light of the findings:

- The boosting model serves as the foundational classifier where the most fantastic accuracy is achieved overall on ISSO-BCQE.
- The other two baseline classifiers could be better in every pre-trained model with a boosting tree.

The correct combination of hyper-parameters may dramatically increase a model's classification rate, which is why hyper-parameters are crucial for achieving strong results when using MLP, CNN and LSTM models. Identifying suitable hyper-parameters, however, is a difficult task that requires grid lookup or haphazard lookup among Cartesian products of hyper-parameter sets. The literature shows that standard SSO optimization performs better than BCQE and grid search methods when finding the ideal hyperparameter. When hyper-parameters are changed in the ISSO optimizer, a model's validation loss often changes gradually. It smoothenes the hyper-parameter surface and forecasts the success of untested hyper-parameters. ID- surface diagram whereby each pre-trained model feature's error classification curve was compared was taken into consideration to represent how well the ISSO optimizer performs visually. Fig 11 shows how the error rate relates to the ten pre-trained model iterations. The charts display the details of the ideal parameter values that the model got. The decision tree classifiers' performance curves on ten pre-trained model feature extraction tasks are shown in Fig 5. The ISSO optimizer and classification error curves are displayed in Fig 11, respectively. Fig 7 shows that the baseline classifiers perform better when the hyper-parameters are adjusted using the ISSO optimizer. The observations point to the following points that should be noted:

- Hyper-parameter adjustment in all baseline classifiers has resulted in a significant increase in classification rate.
- Features from ISSO offer superior accuracy in contrast to other pre-trained models; the decision tree classifier performs better in classification accuracy than the two baseline classifiers.
- The features based on ISSO had the second-best performance when classifying harmful behaviour.

In particular, TOR, Non-TOR, VPN, and Non-VPN are studied in detail with the level-2 classifier used to characterize network traffic in these two distinct ways. Classifying the traffic kinds specified using the feature and output generated by the level-1 classifier is the goal of the level-2 learner. There are a few noteworthy outliers in the case of TOR and Non-TOR, but overall, the bulk class data results display better numbers. Traffic provides a lower accuracy metric when classified as VPN and non-VPN.

4.2. Comparison

This work conducted a series of comparison studies using baseline learning techniques to look at this further. The two classification metrics produced by the six baseline techniques like MLP, CNN, LSTM and LSTM-based CNN are displayed. The average of ten 10-fold cross-validation runs is what is presented in the findings. A categorization method on the second column of the Table organizes the other columns. The two categorization measures are shown in column three and column four. The last row shows the outcomes of the suggested model, while the rows indicate the six baseline techniques. Some figure shows that the proposed model outperforms the traditional models. Promising features that aid in creating an effective prediction model are extracted through pre-trained deep-learning models and time-based feature-to-data transformation. A comparison between the accuracy measure model and the recommended transfer learning model of the outcomes of the cutting-edge technique covered in earlier research is presented. The performance measure is displayed, and the strategies for using time-based features with classifiers are shown in the Table's rows. Bold letters are used to indicate the best values. The presented outcome reveals the suggested model's maximum accuracy values in the network datasets. The last column lists each technique's average performance.

Additionally, it is noted that the suggested model outperforms the existing best performer by 1.97% at its peak performance of 99.6% accuracy. Two primary elements were identified as responsible for the observed improvement in performance: the conversion of numeric values in the data and the extraction of a feature from the pre-trained model through knowledge sharing. Our investigation focused on the theory that has been used intuitively but has never been verified in detecting hazardous traffic, proving that the retrieved feature from the pre-trained model helps the prediction come true. The first suggested method that enhances performance and eliminates the need for the traditional model in predicting using ISSO-BCQE, transforming a time-based characteristic into a sample is known as malicious traffic.

5. Conclusion

The current study developed a bi-level revolutionary learning model to classify the possibly dangerous traffic entries on the network. For each instance of the feature set in the proposed model, the samples created from the matrix are fed into the proposed model to extract the features that describe the various types of traffic. With 96% accuracy rate, the suggested model outperformed the state-of-the-art model which had an accuracy of 99.5%, thanks to the practical application of gradient boosting and pre-trained features. To the best of our knowledge, this is the first research to apply optimization and boosting to categorize malware traffic from network data, according to a literature review. A comparison of similar state-of-the-art methodologies with the proposed model shows that our methods with the learning and boosting model have successfully categorized the patterns of different malware families or the normal traffic from the sample data. The classification accuracy is more remarkable for the suggested model than the conventional machine learning model. Real-time traffic data may be used with the suggested model to detect malware further. In further research, we recommend expanding upon the proposed model. Looking for improved performance, we may consider the deep, surface, and black web from the past few decades. As shown in the literature for comparable domain data, hybridization in the suggested model is another area of future research as a recent paradigm for malware classification.

References

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
- [2] Reddy, Y. Ramadevi, and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICAC)*, Sep. 2016, pp. 1148–1153.
- [3] Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Jan. 2015, pp. 92–96.
- [4] Farnaaz and M. A. Jabbar, "Random forest modelling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, Jan. 2016.
- [5] Khan and N. Jain, "A survey on intrusion detection systems and classification techniques," *Int. J. Sci. Res. Sci., Eng. Technol.*, vol. 2, no. 5, pp. 202–208, 2016.
- [6] Amel Ali Alhussan, Hassan K. Ibrahim Al-Mahdawi, Ammar Kadi, Spam Detection in Connected Networks Using Particle Swarm and Genetic Algorithm Optimization: Youtube as a Case study, *Journal of International Journal of Wireless and Ad Hoc Communication*, Vol. 6 , No. 1 , (2023) : 08-18 (Doi : <https://doi.org/10.54216/IJWAC.060101>)
- [7] Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for an intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.
- [8] Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for an intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.
- [9] Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," *Proc. IEEE Int. Conf. Comput. Sci. Eng./IEEE Int. Conf. Embedded Ubiquitous Comput.*, Jul. 2017, pp. 635–638.
- [10] Sathya Preiya V, Kumar VDA. Deep Learning-Based Classification and Feature Extraction for Predicting Pathogenesis of Foot Ulcers in Patients with Diabetes. *Diagnostics*. 2023; 13(12):1983. <https://doi.org/10.3390/diagnostics13121983>.
- [11] Balakrishnan C, Ambeth Kumar VD. IoT-Enabled Classification of Echocardiogram Images for Cardiovascular Disease Risk Prediction with Pre-Trained Recurrent Convolutional Neural Networks. *Diagnostics (Basel)*. 2023 Feb 18;13(4):775. doi: 10.3390/diagnostics13040775. PMID: 36832263; PMCID: PMC9955174.

- [12] Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring: A survey," Submitted to IEEE Trans. Neural Netw. Learn. Syst., 2016. [Online]. Available: <http://arxiv.org/abs/1612.07640>
- [13] Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn.—Res.*, vol. 11, pp. 3371–3408, 2010.
- [14] Ahmed Mohamed Zaki, Abdelaziz A. Abdelhamid, Abdelhameed Ibrahim, Marwa M. Eid, El-Sayed M. El-Kenawy, Enhancing K-Nearest Neighbors Algorithm in Wireless Sensor Networks through Stochastic Fractal Search and Particle Swarm Optimization, *Journal of Cybersecurity and Information Management*, Vol. 13 , No. 1 , (2024) : 76-84 (Doi : <https://doi.org/10.54216/JCIM.130108>)
- [15] Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl.*, Anaheim, CA, USA, Dec. 2016, pp. 195–200
- [16] Hemamalini, Selvamani, and Visvam Devadoss Ambeth Kumar. 2022. "Outlier Based Skimpy Regularization Fuzzy Clustering Algorithm for Diabetic Retinopathy Image Segmentation" *Symmetry* 14, no. 12: 2512. <https://doi.org/10.3390/sym14122512>.
- [17] Kumar, V.D.A., Sharmila, S., Kumar, A. et al. A novel solution for finding postpartum haemorrhage using fuzzy neural techniques. *Neural Comput & Applic* 35, 23683–23696 (2023). <https://doi.org/10.1007/s00521-020-05683-z>
- [18] V. D. A. Kumar, M. Raghuraman, A. Kumar, M. Rashid, S. Hakak and M. P. K. Reddy, "Green-Tech CAV: Next Generation Computing for Traffic Sign and Obstacle Detection in Connected and Autonomous Vehicles," in *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1307-1315, Sept. 2022, doi: 10.1109/TGCN.2022.3162698.
- [19] Potluri and C. Diedrich, "Accelerated deep neural networks for an enhanced intrusion detection system," *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom.*, Berlin, Germany, Sep. 2016, pp. 1–8.
- [20] Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software-defined networking," *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, Oct. 2016, pp. 258–263
- [21] Hodo, X. J. A. Bellekens, A. Hamilton, C. Tachtatzis, and R. C. Atkinson, Shallow and deep networks intrusion detection system: A taxonomy and survey, Submitted to *ACM Survey*, 2017, [Online]. Available: <http://arxiv.org/abs/1701.02145>
- [22] Kim, G., Yi, H., Lee, J., Paek, Y., Yoon, S.: Lstm-based system-call language modelling and robust ensemble method for designing host-based intrusion detection systems. *arXiv preprint arXiv:1611.01726* (2016)
- [23] Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18(2) (2016) 1153–1176
- [24] Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, New York, NY, USA. Volume 35. (2015) 2126
- [25] Abhishek Kumar, Kamred Udham Singh, Visvam Devadoss Ambeth Kumar, Tapan Kant, Abdul Khader Jilani Saudagar, Abdullah Al Tameem, Mohammed Al Khathami, Muhammad Badruddin Khan, Mozaherul Hoque Abul Hasanat, Khalid Mahmood Malik, " Robust Watermarking Scheme for NIFTI Medical Images", Vol.71, No.2, 2022, pp.3107-3125, doi:10.32604/cmc.2022.022817
- [26] V.D.Ambeth Kumar and M.Ramakrishan (2013), "Temple and Maternity Ward Security using FPRS" in the month of May for the *Journal of Electrical Engineering & Technology (JEET)* , Vol. 8, No. 3, PP: 633-637.
- [28] Safa Otoum, Burak Kantarci, and Hussein T. Mouftah, "Adaptively supervised and intrusion-aware data aggregation for wireless sensor clusters in critical infrastructures," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6
- [29] Arnaldo Gouveia and Miguel Correia, A Systematic Approach for the Application of Restricted Boltzmann Machines in Network Intrusion Detection, vol. 10305, 05 2017.
- [30] Beigh and M. A. Peer, "Performance evaluation of different intrusion detection system: An empirical approach," in *Intl Conf. on Computer Communication and Informatics*, Jan 2014, pp. 1–7.

- [31] Zhou W., Wen J., Koh Y., Xiong Q., Gao M., Dobbie G., Alam S. (2015), "Shilling Attacks Detection in Recommender Systems Based on Target Item Analysis" PLoS One, July, 10(7), p.e0130968
- [32] Kumari T., Bedi P. (2017), "A Comprehensive Study of Shilling Attacks in Recommender Systems", IJCSI International Journal of Computer Science Issues, 14(4), 44-50
- [33] Cao, J., Wu, Z., Mao, B. & Zhang, Y (2013). "Shilling attack detection utilizing semi-supervised learning method for attack detection utilizing semi-supervised learning method for collaborative recommender system". World Wide Web Journal, 16(5-6): 729-748.
- [34] Yu H, Gao R, Wang K, Zhang F (2017), "A novel robust recommendation method based on kernel matrix factorization". J Intell Fuzzy Syst 32(3):2101–2109
- [35] Yang Z., Cai Z. (2017), "Detecting abnormal profiles in collaborative filtering recommender systems". Journal of Intelligent Information Systems, 48(3), 499-518
- [36] Zhou W., Wen J., Qu Q., Zeng J., Cheng T. (2018), "Shilling attack detection for recommender systems based on the credibility of group users and rating time series", PLoS One, May, 13(5), p.e0196533
- [37] Turk A., Bilge A., (2019). "Robustness analysis of multi-criteria collaborative filtering algorithms against shilling attacks", Expert Systems with Applications, 115, p.386-402
- [38] Moradi P., Ahmadian S., (2015), "A reliability-based recommendation method to improve trust-aware recommender systems", Expert Systems with Applications, 42, 7386-7389.
- [39] Paradarami, N.D. Bastian, J.L. Wightman, "A Hybrid recommender system using artificial neural networks", Expert Systems with Applications, Vol. 83, (2017), 300-313.
- [40] Agar ap, "A neural network architecture combines gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," Proc. 10th Int. Conf. Mach. Learn. Comput., Feb. 2018, pp. 26–30.
- [41] Around, M.-A. El Hussaini, A. El Hore, and J. Ben-Othman, "Real-time detection of MAC layer misbehaviour in mobile ad hoc networks," Appl. Comput. Information., vol. 13, no. 1, pp. 1–9, 2017.
- [42] P. Sherubha, P Amudhavalli, SP Sasirekha, "Clone attack detection using random forest and multi-objective cuckoo search classification", International Conference on Communication and Signal Processing (ICCSP), pp. 0450-0454, 2019.
- [43] S. Dinesh, K. Maheswari, B. Arthi, P. Sherubha, A. Vijay, S. Sridhar, T. Rajendran, and Yosef Asrat Waji, "Investigations on Brain Tumor Classification Using Hybrid Machine Learning Algorithms", Hindawi Journal of Healthcare Engineering, Volume 2022.