SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I10P120

Volume 11 Issue 10, 247-257, October 2024

Original Article

Enhancing Cyber Security Via Malware Classification using Tuna Swarm-Based Feature Selection with Optimal Deep Learning

V. S. Pavankumar[1], S. Arivalagan[2], M. Murugesan[3], P. Sudhakar[4]

[1,2,4]*Department of Computer Science and Engineering, Annamalai University, Tamilnadu, India.*
[3]*Department of Computer Science and Engineering, Anurag Engineering College, Telangana, India.*

[1]*Corresponding Author : pavankumar620@gmail.com*

Received: 20 August 2024 Revised: 21 September 2024 Accepted: 20 October 2024 Published: 30 October 2024

Abstract *- Malware detection is a central aspect of cyber security that includes detecting and mitigating malicious software, often called malware, that can compromise the safety and integrity of a computer system. Different malware detection techniques are used, such as Machine Learning (ML), signature-based detection, and behavioural analysis. Cutting-edge ML models are widely applied for malware detection. These techniques analyze large datasets to detect features and patterns related to malicious behaviours. Supervised learning trains models on labelled datasets, while unsupervised learning can identify anomalies in system behaviours without predefined labels. Deep Learning (DL)-based malware detection improves the capability to identify polymorphic and sophisticated risks and promotes a more adaptive proactive cyber security system. This study introduces Malware Recognition and Classification using the Tuna Swarm Optimization-based Feature Selection with DL (MRC-TSOFSDL) approach. In the MRC-TSOFSDL approach, the feature subset selection process is accomplished using the TSO model. The Stacked Sparse Autoencoder (SSAE) method is used to recognise malware automatically. Chimp optimization Algorithm (ChoA) based on a hyperparameter tuning process is utilized to improve the malware detection outcomes of the SSAE model. The performance analysis of the MRC-TSOFSDL method is examined under a malware dataset. The comparative results of the MRC-TSOFSDL technique demonstrated a maximum accuracy value of 98.65% over existing models.*

Keywords *- Malware detection, Cybersecurity, Deep Learning, Tuna Swarm Optimization, Feature selection.*

1. Introduction

The Internet plays a significant role in data transfer among various nodes. It comprises many devices, networks, and computers [1]. Accordingly, the Internet has become the objective of cybercriminals. Examples of cyber-attacks are malware, spam, and phishing. Malware is categorized as a primary security attack in cyberspace. Malware defines the integration of 'mal' from 'malicious' and 'ware' from 'software'. It is a code element privately implanted into a computer network or system maliciously intended to disturb the normal flow of events [2]. According to its functionality, malware can be rarely categorized into Trojan horses, worms, and viruses. Cybercriminals employ different obfuscation techniques for producing malware variants. These techniques include malicious challenges into binary and textual information that malware detectors determine the complexity for interpretation and detection [3].

The malicious pattern can be located within a software application or data file. These software applications can involve various platforms like Android, Linux, and Windows. Malware detection research employing ML is developing in popularity because it is a practical approach that could produce a higher level of detection accuracy [4]. A few earlier studies employed ML methods that will help make decisions once they learn from the data patterns. ML defines the abstraction of decreasing human involvement in computational systems [5]. ML predicts the decisions with the help of computer learning techniques and involvements or prior data [6].

The unsupervised and supervised learning techniques are employed to examine the features and monitor the model. In both scenarios, the machine could be learned to differentiate between benign and malicious activities. During the supervised learning, the ML method was provided with the input and targets and also learned to constantly the regular activities with the "normal" classes relate and relate real malware patterns with their respective "malware" types [7]. The training method is frequent until the technique learns to anticipate each sample appropriately. Several ML techniques are employed to design malware detection methods [8].

Unsupervised learning techniques offer inputs without any objectives, and the ML technique will learn to differentiate between benign and malware instances.

Moreover, a few researches combined the unsupervised and supervised learning methods [9]. Malware detection is a significant security aspect that has a robust relationship with an organization's reputation and economic and legal issues. The DL technique is utilized to create and fix detection mechanisms to better resolve several difficulties with identifying the malware [10].

This study introduces Malware Recognition and Classification using the Tuna Swarm Optimization-based Feature Selection with DL (MRC-TSOFSDL) approach. In the MRC-TSOFSDL approach, the feature subset selection process is accomplished using the TSO model. The Stacked Sparse Autoencoder (SSAE) method is used to automate the recognition of malware. Chimp optimization Algorithm (ChoA) based on a hyperparameter tuning process is utilized to improve the malware detection outcomes of the SSAE model. The performance analysis of the MRC-TSOFSDL method is examined under a malware dataset. The key contribution of the MRC-TSOFSDL technique is listed below.

- The MRC-TSOFSDL model utilizes the TSO technique to enhance Feature Selection (FS) in the malware recognition process, confirming that the most relevant features are prioritized. This optimization method improves the overall accuracy of malware detection. By effectually choosing features, the methodology mitigates computational complexity while maintaining high performance.
- The MRC-TSOFSDL methodology employs the SSAE approach to automate the recognition and identification of malware, enabling the effective processing of complex data. This model improves the capability to detect subtle patterns indicative of malicious behaviour. Utilizing DL enhances accuracy and mitigates the time needed for manual analysis.
- The MRC-TSOFSDL model incorporates ChoA for hyperparameter tuning, which substantially improves the accomplishment of the SSAE model. This optimization confirms that the model operates at its best, adapting to the complexities of the data. By fine-tuning parameters effectively, the methodology achieves enhanced detection outcomes in malware classification.
- The MRC-TSOFSDL approach incorporates TSO for FS with advanced AE architectures and optimization techniques, creating a more effectual and efficient framework for malware detection. The novelty is integrating these diverse methodologies, which concurrently improves feature relevance and model performance. This overall strategy allows for better handling of intrinsic data patterns, setting it apart from conventional malware detection methods.

2. Literature Survey

Borra et al. [11] presented the Optimized Email Classification Network (OEC-Net) architecture. This technique integrates a Principal Component Analysis (PCA) method for extraction. The OEC-Net employs compelling features of PSO to combine a DL Convolutional Neural Networks (DLCNNs) method for classification. In [12], an innovative DEEPSEL (Deep FS) method was introduced, a DL-based method for recognizing malware and malicious programs. DEEPSEL utilizes a group of aspects to describe the Android version and categorizes them as authentic and malevolent. The deployment of PSO was considered to significantly impact executing FS. In [13], a higher-performance malware detection method employing DL and FS techniques has been presented. Two diverse malware datasets could be used to identify and differentiate the malware from benign activities. The databases were pre-processed, followed by correlation-based FS, which was implemented to generate various FS datasets. The dense and Long Short-Term Memory (LSTM)-based DL methods were trained by employing several FS databases.

In [14], a Deep Hashing (DH)-based malware classifying method was developed that comprises two parts: ResNet50-based DH for malware classification. Numerous DH methods have been presented, such as extracting the higher-layer outputs at the ResNet50. Likewise, a ResNet50-based Deep Polarized Network (RNDPN) was developed for returning similar samples to Top K. In the secondary portion, a Hamming-distance- and majority-based voting could be developed for malware recognition based on the retrieved outcomes. Akhtar and Feng [15] employed a hybrid DL method, such as the CNN-LSTM method. A revolutionary DL method was introduced.

Consequently, automated extraction of higher-level representations and abstractions supports the malware classification method. Mokkapati and Dasari [16] employ the ANN-assisted Intellectual Non-Dependent FS Method (ANN-INDFSM) technique. The ANN-based FS technique has been utilized to remove non-salient features and find dimensionality ranges. Numerous techniques are employed for FS and extraction. The effectiveness of the ML method could be enhanced by eliminating redundant and unnecessary features.

Qamar [17] presented new methods in cyber safety prevention related to Electronic Health Records (EHR) in FS and classification employing DL techniques. The input EHR data was handled to remove noise and eliminate null values. The processed data was chosen based on their aspects using a kernel-assisted gradient. Rahima Manzil and Naik [18] introduce a Hamming distance-based FS methodology. The model also utilizes a Python tool. Fiza et al. [19] propose a model using an Improved COA (ICOA) for FS and a Deep Neural Network Framework (DNNF). Polatidis et al. [20]

proposed an effective FS model. A pre-processing approach is presented to choose optimal features, confirming high accuracy in malware detection.

Chaganti, Ravi, and Pham [21] present the DL CNN method using a fusion feature set model. Mittal and Pandian [22] aim to evaluate the efficiency of ML and DL techniques, specifically CNN and RNN. Ravi and Alazab [23] incorporate a multi-head attention mechanism and a CNN model. Akuthota and Bhargava [24] propose a unique technique incorporating ensemble feature extraction methods with LSTM networks. Shaukat, Luo, and Varadharajan [25] introduce a DL model for malware detection, which extracts deep features with a fine-tuned model and utilizes SVM for detection, minimizing the requirement for feature engineering and domain knowledge. Kumar, Janet, and Neelakantan [26] present an Intelligent Malware Classification Method (IMCNN) that utilizes deep CNNs. The technique also utilizes pre-trained models, namely VGG16, VGG19, InceptionV3, and ResNet50 and visualizes malware as grayscale images. Features are extracted using ReLU, PCA, and SVD, with k-NN, Random Forest (RF), and SVM. Jebin Bose and Kalaiselvi [27] employ the Metaheuristic Artificial Jellyfish Search Optimizer (MH-AJSO) and classify malware types with the DDResNet101 classifier. The proposed architecture uses a feature extraction technique improved by an optimization approach to choose key features but may lose crucial data, affecting accuracy. DL techniques for malware recognition may need help with generalization to new variants, while detection models can face scalability issues and overlook interactions among features. Furthermore, hybrid models integrating diverse architectures might be computationally intensive, and static analysis methods may miss dynamic ransomware behaviours. Despite enhancements in malware detection using DL and FS techniques, many models still struggle with generalization to new malware variants and often depend on static analysis, which can miss dynamic behaviours. Moreover, there is a requirement for more effectual approaches that minimize feature engineering while maintaining high accuracy and adaptability across diverse datasets. Furthermore, existing models may not efficiently incorporate various feature extraction techniques to improve classification performance comprehensively.

3. The Proposed Method

This study introduces a novel MRC-TSOFSDL model. The technique involves the concept of FS with an optimum hyperparameter selection strategy for enhancing the detection results of the malware recognition process. It contains three significant processes involving TSO-based FS, SSAE-based classification, and ChoA-based tuning. Figure 1 establishes the complete structure of the MRC-TSOFSDL method.

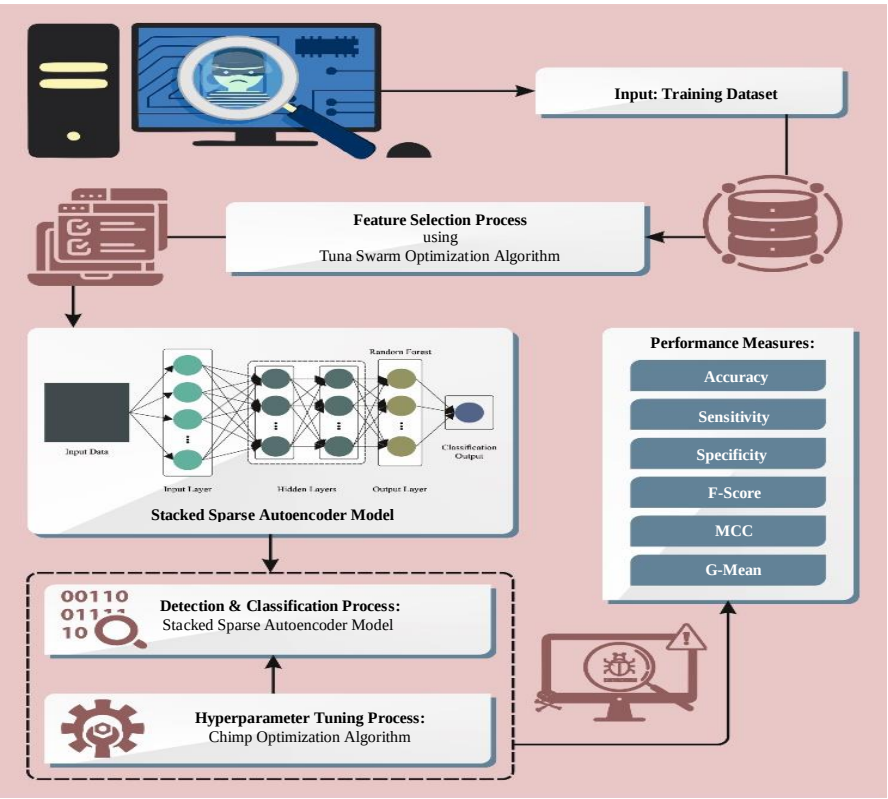

Fig. 1 Overall flow of the MRC-TSOFSDL approach

3.1. FS using the TSO Model

Initially, the MRC-TSOFSDL technique undergoes the feature subset selection process using the TSO technique [28]. This model is chosen for FS because it can effectively explore complex search spaces and detect optimal feature subsets. Its biologically inspired approach replicates the cooperative behaviour of tuna, resulting in robust solutions that improve the model's performance. Compared to other techniques, TSO presents greater convergence speed and accuracy, making it specifically effective for dynamic datasets in malware detection.

Generally, the tuna is said to be a social animal. They usually pick the equivalent predation plan based on the objective they are searching for. Spiral foraging is the initial plan; when tuna serve, they dip into a spiral to capture their food in surface water, then attack and hold it. Parabolic foraging is the next strategy in which every tuna monitors the preceding individual to create an illustrative outline to encircle its target. Tuna effectively hunts over the above dual models. TSO depends on this normal foraging behaviour, and the system follows the simple guidelines below.

3.1.1. Population Initializes

TSO begins an optimizer procedure by creating the first swarm at random uniformly to upgrade the location.

$$X^{ini} = rand \cdot (ub - lb) + lb, i = 1,2, \dots, NP \quad (1)$$

X^{ini} denotes the initial populace; lb and ub signify the lower and upper bounds.

3.1.2. Spiral Foraging

Generally, the tuna hunts their target by creating a fit spiral and hunt their target. In addition, they trade data with each other. Every tuna monitors the preceding one, so data is dispersed among nearby tuna. Depending upon the above standards, the formulations of the spiral foraging tactic are defined below:

$$X_i^{t+1} = \begin{cases} \alpha_1 \cdot (X_{best}^t + \beta \cdot |X_{besi}^t - X_i^t|) + \alpha_2 \cdot X_i^t, & i = 1 \\ \alpha_1 \cdot (X_{best}^t + \beta \cdot |X_{besi}^t - X_i^t|) + \alpha_2 \cdot X_{i-1}^t, & i = 2,3, \dots, NP \end{cases} \quad (2)$$

$$\alpha_1 = a + (1 - a) \cdot \frac{t}{t_{\max}} \quad (3)$$

$$\alpha_2 = (1 - a) - (1 - a) \cdot \frac{t}{t_{\max}} \quad (4)$$

$$\beta = e^{bl} \cdot \cos(2\pi b) \quad (5)$$

$$l = \exp\left(3 \cdot \cos\left(\left(\frac{t_{\max}+1}{t} - 1\right)\right) \cdot \pi\right) \quad (6)$$

Where α_1 and α_2 signify the co-efficient that manages the individual near the optimum and preceding in the chain, X_{best}^t represents the present optimum individual; a refers to the constant, which is 0.6, t and $t_{\max}$ denotes the existing and maximum iterations count, correspondingly, X_i^{t+1} denotes the location of the ith individual in the $t + 1$ generation, and b refers to the random number that uniformly spreads from $zero$ to one. When an optimal individual finds it hard to locate food, it follows the best individual, creating a random coordinate as a position for a spiral search.

This allows every individual to find a larger area, enabling TSO to traverse the search space effectually. The mathematical method is defined as follows:

$$X_i^{t+1} = \begin{cases} \alpha_1 \cdot (X_{rand}^t + \beta \cdot |X_{rand}^t - X_i^t|) + \alpha_2 \cdot X_i^t, & i = 1 \\ \alpha_1 \cdot (X_{rand}^t + \beta \cdot |X_{rand}^t - X_i^t|) + \alpha_2 \cdot X_{i-1}^t, & i = 2,3, \dots, NP \end{cases} \quad (7)$$

Metaheuristics normally execute a wide global search in the initial phase, monitored by a gradual change to exact local growth. So, as iterations count upsurges, TSO swaps the position point from arbitrary to optimum individuals for spiral foraging. Overall, the last mathematical method of the spiral foraging tactic is defined below:

$$X_i^{t+1} = \begin{cases} \alpha_1 \cdot (X_{rand}^t + \beta \cdot |X_{rand}^t - X_i^t|) + \alpha_2 \cdot X_i^t, & i = 1 \\ \alpha_1 \cdot (X_{rand}^t + \beta \cdot |X_{rand}^t - X_i^t|) + \alpha_2 \cdot X_{i-1}^t, & i = 2,3, \dots, NP' \\ \alpha_1 \cdot (X_{best}^t + \beta \cdot |X_{best}^t - X_i^t|) + \alpha_2 \cdot X_i^t, & i = 1 \\ \alpha_1 \cdot (X_{best}^t + \beta \cdot |X_{best}^t - X_i^t|) + \alpha_2 \cdot X_{i-1}^t, & i = 2,3, \dots, NP' \end{cases} \quad \begin{matrix} if\ rand < \frac{t}{t_{\max}} \\ \\ if\ rand \geq \frac{t}{t_{\max}} \end{matrix} \quad (8)$$

3.1.3. Parabolic Foraging

Additionally, tuna generate a parabolic cooperative feeding creation to feed over a spiral creation. The tuna is served in a parabolic outline utilizing prey as a position point. Also, the tuna can search for prey by watching everywhere. Let's consider that the selection probability is 50 percent, and both models have been performed simultaneously. Tuna hunt coactively over these two foraging tactics and then discover their target.

$$X_i^{t+1} = \begin{cases} X_{best}^t + rand \cdot (X_{best}^t - X_i^t) + TF \cdot p^2 \cdot (X_{best}^t - X_{best}^t), & if\ rand < 0.5 \\ TF \cdot p^2 \cdot X_i^t, & if\ rand \geq 0.5 \end{cases} \quad (9)$$

$$p = \left(1 - \frac{t}{t_{max}}\right)^{\left(\frac{t}{t_{\max}}\right)} \quad (10)$$

In this formulation, TF denotes the randomly generated number within [1, -1].

3.1.4. Condition of Termination

The TSO technique constantly upgrades and computes all separate tuna until the last condition is detected, giving back the equivalent fitness values and best individual. FS aims to enhance classification performance by choosing the best optimum feature set [29]. A trimmed database delivers quicker training time and better accuracy. So, a Fitness Function (FF) is utilized, which fulfils goals such as increasing classification accuracy and reducing features. The FS model is deliberated in this framework by employing the TSO technique.

$$F = \max\left(Acc + wf\left(1 - \frac{F_s}{F_t}\right)\right) \quad (11)$$

Here, wf denotes the weight factor, Acc signifies the accuracy, F_s represents the length of nominated features and F_t refers to the overall feature number. The weight factor value is set nearer to one, representing that both objectives, such as feature minimization and accuracy enhancement, are similarly considered vital.

3.2. SSAE-Based Classification Process

This work applies the SSAE model to recognize and identify malware automatically [30]. The SSAE is ideal as it learns hierarchical representations, capturing intrinsic patterns indicative of malicious behaviour. Its capability to mitigate dimensionality while conserving crucial features is significant for handling large datasets. The sparsity constraint concentrates on relevant features, improving discriminative power. Furthermore, the stacked architecture enables more profound learning, allowing for better generalization across various malware variants and making SSAEs a superior choice to conventional methods in accuracy and robustness. An SAE is a typical 3-layered artificial neural network (ANN), which uses unsupervised learning to acquire input. The compressed code captures the reduced dimensions of the raw input, with AE serving as an effective feature extractors for DL models. Typically, an SAE consists of the decoder ϕ and encoder ϕ

$$\varphi: Z = \sigma(WX + b) \quad (12)$$

$$\phi: X' = \sigma'(WZ + b') \quad (13)$$

Whereas R^M refers to the compressed code. $\sigma, W, b,$ and σ', W', b' denote the encoding and decoding process's corresponding activation function, weight coefficient matrices, and bias vector. $Z \in X' \in R^N$ denotes the vector of reconstruction. $X \in R^N$ refers to the input. The SAE converts the input X into hidden variable Z, and then rebuild X' over the decoder. So, the optimizer AE trains W' and W. This enables the decoder output to closely approximate the original input. For an assumed data set X, the reconstruction error is defined below:

$$L(X, X') = \|X, X'\|^2 = \|X - \sigma'[W'\sigma(WX + b) + b']\|^2 \quad (14)$$

The stacked self-encoder system is an NN that contains numerous AE layers that use the prior output of the AE layer as an input for the next one. For an assumed training sample $X^{(i)}$ set, i represents the sample count. The loss function for the SAE is stated in the below-mentioned method:

$$J(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{2}\left\|\sigma(WX^{(i)} + b) - Y^{(i)}\right\|^2\right] + \frac{\lambda}{2}\sum_{l=1}^{n-1}\sum_{i=1}^{s_l^c}\sum_{j=1}^{s_l^r}\left[W_{ji}^{(L)}\right]^2 \quad (15)$$

Here, m specifies the total samples, and Y depicts the equivalent X label. The 2nd term is the weight reduction regularization term, employed to avert overfitting. λ denotes the weight reduction co-efficient, which alters the relative weight of the 1st and 2nd term. W_{ji}^{l} signifies the connection weight among neurons in dual layers. s_l^r and s_l^c represent the total neuron count in layer l. The sparsity factors can be attained by inserting terms in Equation (15), and the entire feature extractor procedure is improved. Then, the SSAE loss function is conveyed as follows:

$$L_{sparse}(W, b) = J(W, b) + \beta\sum_{j=1}^{s_2} KL\left(\rho\hat{\rho}_j\right) \quad (16)$$

$$KL\left(\rho\hat{\rho}_j\right) = \rho\log\frac{\rho}{\hat{\rho}_j} + (1 - \rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \quad (17)$$

While $\hat{\rho}$ represents the value of mean activation, ρ and β refer to the sparseness and divergence constant, correspondingly, and s_2 denotes the number of HL neurons.

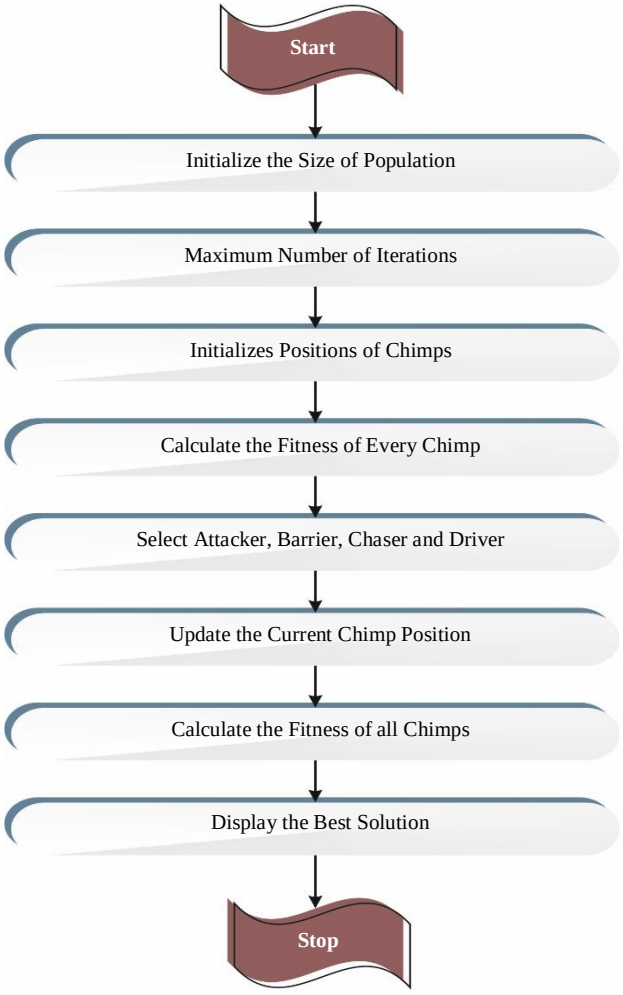

Fig. 2 Flowchart of ChoA

3.3. Hyperparameter Tuning

Finally, the ChoA-based hyperparameter tuning procedure is accomplished to enhance the malware detection outputs of the SSAE method. This model is chosen for its effectual balance between searching for new solutions and refining existing ones, enabling it to navigate complex parameter spaces efficiently. Its unique evolutionary approach replicates the behaviour of chimpanzees, resulting in enhanced convergence rates and optimal solutions. Compared to conventional tuning methodologies, ChoA improves the model's performance by systematically refining hyperparameters, resulting in more accurate and robust malware detection. Figure 2 illustrates the flowchart of ChoA.

This technique appeals to motivation from chimpanzee cleverness and breeding behaviour in group hunting [31]. Equation (18) signifies the distance (D) between the chimpanzee and its target, whereas Equation (19) denotes the chimpanzee's location upgrade formulation; α_{prey} refers to the vector of prey locations and α_{chimp} signifies the chimpanzee's position vector.

$$D = \left|C \cdot \alpha_{prey} - m \cdot \alpha_{chimp}\right| \tag{18}$$

$$\alpha_{chimp}(n+1) = \left|\alpha_{prey} - a \cdot d\right| \tag{19}$$

The constant vectors are m, c, and d and are expressed by Equation (20) through (22).

$$a = 2 \cdot l \cdot r_1 - 1 \tag{20}$$

$$c = 2 \cdot r_2 \tag{21}$$

$$m = chaotic\ value \tag{22}$$

Whereas l denotes a static value that droplets beside a line from 2.5 to 0 through the iterations, r_1 and r_2 are said to be arbitrary values among (0,1), and m signifies the chaotic vector. The Attacker, Driver, Chaser, and Barrier (ADCB) are the finest results with optimum abilities for arithmetically pretending to use this method. C denotes the arbitrary variable that affects the location of the victim within [0,2] on the specific place of chimpanzees (when $C < 1$ and $C > 1$, the grade of influence weakens and strengthens, respectively).
The locations of other chimpanzees in the populace are defined by the locations (d) of ADCB and the location upgrade Equations (23), (24), (25), and (26).

$$d_{attack} = \left|C_1 \cdot \alpha_{attacker} - m_1 \cdot x_n\right| \tag{23}$$

$$d_{barrier} = \left|C_2 \cdot \alpha_{barrier} - m_2 \cdot x_n\right| \tag{24}$$

$$d_{chaser} = \left|C_3 \cdot \alpha_{chaser} - m_3 \cdot x_n\right| \tag{25}$$

$$d_{driver} = \left|C_4 \cdot \alpha_{driver} - m_4 \cdot x_n\right| \tag{26}$$

Where α signifies the four chimp positions vector. Ensuing this, the chimpanzees' following points $(x_1, x_2, x_3\ and\ x_4)$ are modernized by employing Equation (27) through (30):

$$x_1 = \alpha_{attacker} - a_1 \cdot d_{attacker} \tag{27}$$

$$x_2 = \alpha_{barrier} - a_2 \cdot d_{barrier} \tag{28}$$

$$x_3 = \alpha_{chaser} - a_3 \cdot d_{chaser} \tag{29}$$

$$x_4 = \alpha_{driver} - a_4 \cdot d_{driver} \tag{30}$$

The positions are updated by utilizing Equation (31):

$$x_{chimp} = \frac{x_1 + x_2 + x_3 + x_4}{4} \tag{31}$$

Then, Equation (32) is executed once the locations are updated.

$$\alpha_{chimp}(n+1) = \begin{cases} \alpha_{prey} - x \cdot d, \emptyset < 0.5 \\ chaotic\ , \emptyset < 0.5 \end{cases} \tag{32}$$

Fitness selection is a crucial aspect that affects the accomplishment of the ChoA technique. The optimization process comprises a solution encoding model to evaluate the efficiency of potential candidate outputs. Also, the ChoA method regards accuracy as the vital principle for projecting the FF, which can be expressed below.

$$Fitness\ =\ \max(P) \tag{33}$$

$$P = \frac{TP}{TP + FP} \tag{34}$$

From the above equation, TP and FP indicate the true and false positive values.

4. Experimental Validation

The malware detection outputs of the MRC-TSOFSDL approach are investigated using a malware database [32] comprising 10868 instances with nine classes, as described in Table 1.

Table 1. Dataset specification

Labels	Class	Instance Numbers
C-1	Ramnit	1541
C-2	Lollipop	2478
C-3	Kelihos_ver3	2942
C-4	Vundo	475
C-5	Simda	42
C-6	Tracur	751
C-7	Kelihos_ver1	398
C-8	Obfuscator.ACY	1228
C-9	Gatak	1013
Total Number of Instances		**10868**

Figure 3 depicts the confusion matrices of the MRC-TSOFSDL method under 80:20 and 70:30 of TRAPH/TESPH. The experimentations indicate the improved outcome with nine classes.

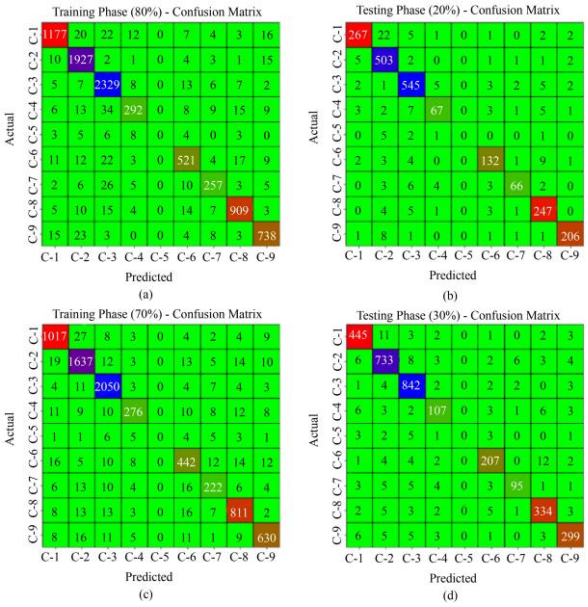

Fig. 3 Confusion matrices under (a-b) 80:20, and (c-d) 70:30 of TRAPH/TESPH.

Table 2. The malware detection output of the MRC-TSOFSDL approach under 80:20 of TRAPH/TESPH

Labels	$Accu_y$	$Sens_y$	$Spec_y$	F_{score}	MCC	G_{mean}
TRAPH (80%)						
C-1	98.38	93.34	99.23	94.35	93.41	94.35
C-2	98.48	98.17	98.57	96.69	95.72	96.70
C-3	97.95	97.98	97.94	96.32	94.93	96.33
C-4	98.45	75.65	99.51	81.22	80.66	81.45
C-5	99.67	12.12	100.00	21.62	34.76	34.82
C-6	98.37	86.98	99.21	88.01	87.14	88.01
C-7	98.87	81.85	99.51	83.99	83.43	84.02
C-8	98.73	94.00	99.33	94.29	93.58	94.30
C-9	98.68	92.95	99.25	92.77	92.04	92.77
Average	**98.62**	**81.45**	**99.17**	**83.25**	**83.96**	**84.75**
TESPH (20%)						
C-1	98.80	95.36	99.31	95.36	94.67	95.36
C-2	98.16	97.67	98.31	96.18	94.98	96.19
C-3	97.61	96.46	98.01	95.45	93.83	95.45
C-4	98.44	75.28	99.42	79.76	79.10	79.90
C-5	99.59	00.00	100.00	00.00	00.00	00.00
C-6	98.39	86.84	99.26	88.29	87.44	88.31
C-7	98.85	78.57	99.67	84.08	83.70	84.28
C-8	98.16	94.64	98.64	92.51	91.49	92.53
C-9	99.03	94.06	99.59	95.15	94.62	95.16
Average	**98.56**	**79.88**	**99.14**	**80.75**	**79.98**	**80.80**

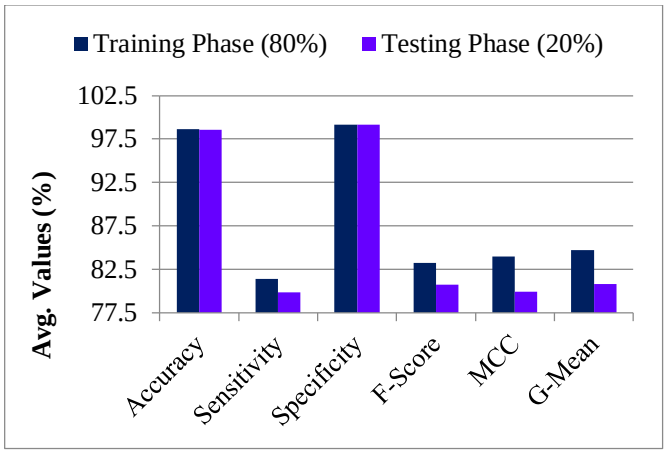

Fig. 4 Average of the MRC-TSOFSDL methodology under 80:20 of TRAPH/TESPH

Table 3. Malware detection outputs of the MRC-TSOFSDL methodology under 70:30 of TRAPH/TESPH

Labels	$Accu_y$	$Sens_y$	$Spec_y$	F_{score}	MCC	G_{mean}
TRAPH (70%)						
C-1	98.29	94.69	98.88	93.99	93.00	94.00
C-2	97.75	95.56	98.39	95.04	93.59	95.04
C-3	98.48	98.27	98.55	97.25	96.20	97.25
C-4	98.66	80.23	99.53	84.40	83.83	84.52
C-5	99.66	00.00	100.00	00.00	00.00	00.00
C-6	97.96	85.16	98.90	85.08	83.99	85.08
C-7	98.61	79.00	99.36	80.73	80.03	80.75
C-8	98.32	92.90	99.02	92.69	91.74	92.69
C-9	98.55	91.17	99.29	91.97	91.18	91.97
Average	**98.48**	**79.67**	**99.10**	**80.13**	**79.28**	**80.14**
TESPH (30%)						
C-1	98.47	95.29	99.00	94.68	93.79	94.68
C-2	97.82	95.82	98.44	95.38	93.96	95.38
C-3	98.50	98.36	98.54	97.17	96.16	97.18
C-4	98.68	81.68	99.39	83.27	82.60	83.28
C-5	99.54	06.25	100.00	11.76	24.94	25.00
C-6	98.62	89.22	99.34	90.20	89.46	90.20
C-7	99.02	81.20	99.68	85.59	85.21	85.71
C-8	98.53	94.08	99.07	93.30	92.47	93.30
C-9	98.68	92.86	99.32	93.29	92.56	93.29
Average	**98.65**	**81.64**	**99.20**	**82.74**	**83.46**	**84.23**

A wide-ranging malware detection analysis of the MRC-TSOFSDL method can be reported with 70:30 of TRAPH/TESPH, as shown in Table 3 and Figure 5. The experimental values denote the improved malware detection outcomes of the MRC-TSOFSDL technique. Based on 70% of TRAPH, the MRC-TSOFSDL technique attains average $accu_y$, $sens_y$, $spec_y$, F_{score}, MCC, and G_{mean} of 98.48%,

79.67%, 99.10%, 80.13%, 79.28%, and 80.14%, respectively. Moreover, with 30% of TESPH, the MRC-TSOFSDL technique gains average $accu_y$, $sens_y$, $spec_y$, F_{score}, MCC, and G_{mean} of 98.65%, 81.64%, 99.20%, 82.74%, 83.46%, and 84.23%, respectively.

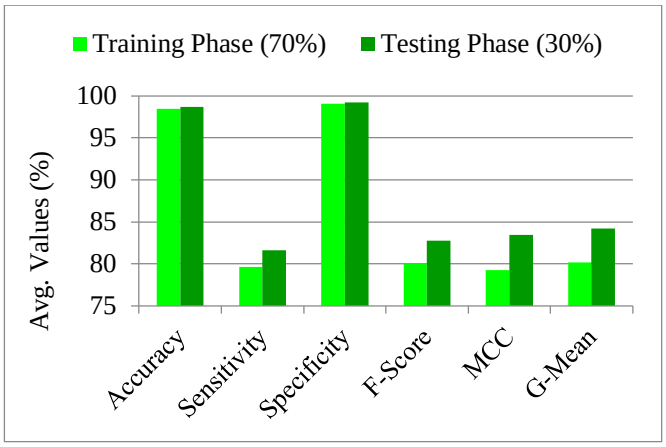

Fig. 5 Average of the MRC-TSOFSDL methodology under 70:30 of TRAPH/TESPH

The performance of the MRC-TSOFSDL approach with 70:30 of TRAPH/TESPH is shown in Figure 6 under Training Accuracy (TRAA) and Validation Accuracy (VALA). This figure illustrates the evaluation of the MRC-TSOFSDL technique over various epochs, showing continuous improvement in TRAA/VALA and confirming its effectiveness.

The improved results in VALA exhibit the MRC-TSOFSDL method's capability to change to the TRA data and better classify unseen data precisely, underscoring the robust generalization capabilities.

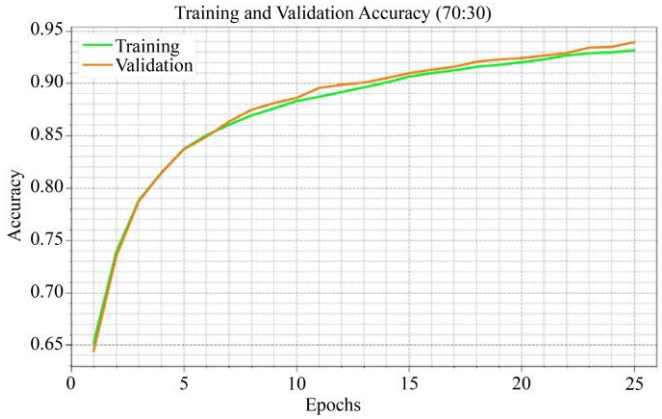

Fig. 6 $Accu_y$ curve of the MRC-TSOFSDL methodology under 70:30 of TRAPH/TESPH

Figure 7 displays the Training Loss (TRLA) and Validation Loss (VALL) outputs of the MRC-TSOFSDL approach under 70:30 of TRAPH/TESPH under diverse epochs. The steady decrease in TRLA reflects the MRC-TSOFSDL approach, optimizing weights and minimizing errors in TRA/TES data, while the figure highlights its efficiency in pattern detection. Significantly, the MRC-TSOFSDL method continuously improves its parameters in lesser discrepancies among the anticipation and real TRA classes.

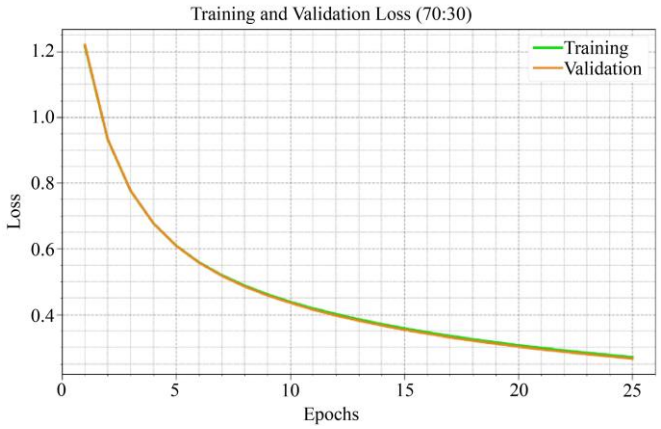

Fig. 7 Loss curve of the MRC-TSOFSDL methodology under 70:30 of TRAPH/TESPH

Figure 8 highlights the enhanced PR values of the MRC-TSOFSDL technique with 70:30 TRAPH/TESPH. This examines the enhanced capabilities of the MRC-TSOFSDL technique in detecting diverse classes and displaying proficiency in class detection.

Similarly, in Figure 9, the MRC-TSOFSDL technique with 70:30 of TRAPH/TESPH exhibited improved ROC outputs in categorizing multiple labels.

It highlights the trade-off between TPR/FRP across thresholds and epochs, demonstrating the MRC-TSOFSDL approach's improved classifier outputs and effectiveness in classification challenges.

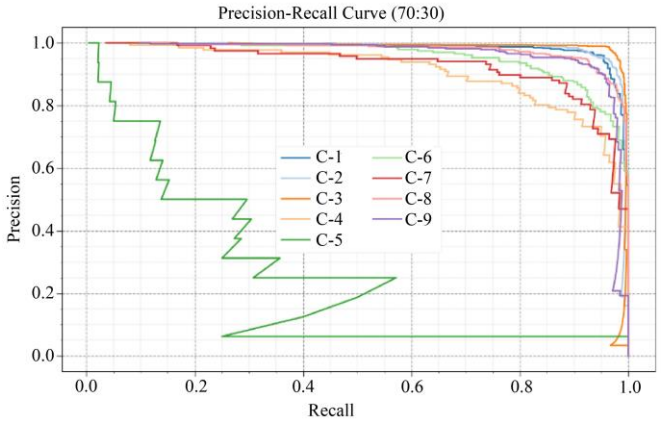

Fig. 8 PR curve of the MRC-TSOFSDL methodology under 70:30 of TRAPH/TESPH

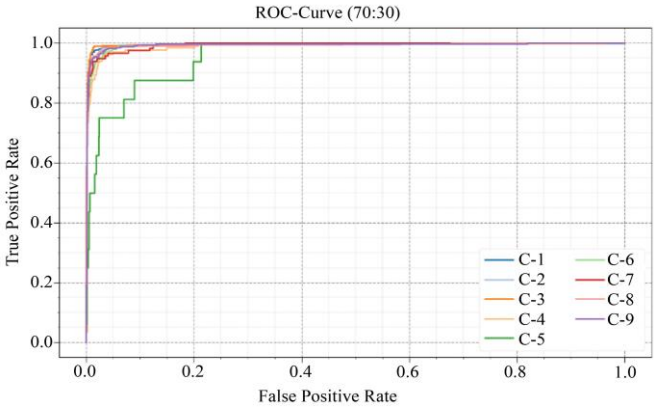

Fig. 9 ROC curve of the MRC-TSOFSDL methodology under 70:30 of TRAPH/TESPH

The $accu_y$ and Computational Time (CT) outputs of the MRC-TSOFSDL method compared to recent techniques is shown in Table 4 [33]. The $accu_y$ results of the MRC-TSOFSDL method with recent approaches are exhibited in Figure 10. These results indicate that the RBF-SVM, CNN, and DLMD methods obtain reduced $accu_y$ values of 93.82%, 95.62%, and 95.21%, respectively. Although linear SVM and MLP methods accomplish closer $accu_y$ values of 96.82% and 97.18%, the MRC-TSOFSDL technique surpasses the others with an increased $accu_y$ of 98.65%.

Table 4. Accuracy and CT outputs of the MRC-TSOFSDL model with other techniques

Methods	Accuracy (%)	CT (sec)
MRC-TSOFSDL	98.65	5.09
Linear-SVM	96.82	14.23
RBF-SVM	93.82	12.69
MLP	97.18	18.2
CNN	95.62	9.62
DLMD	95.21	18.42

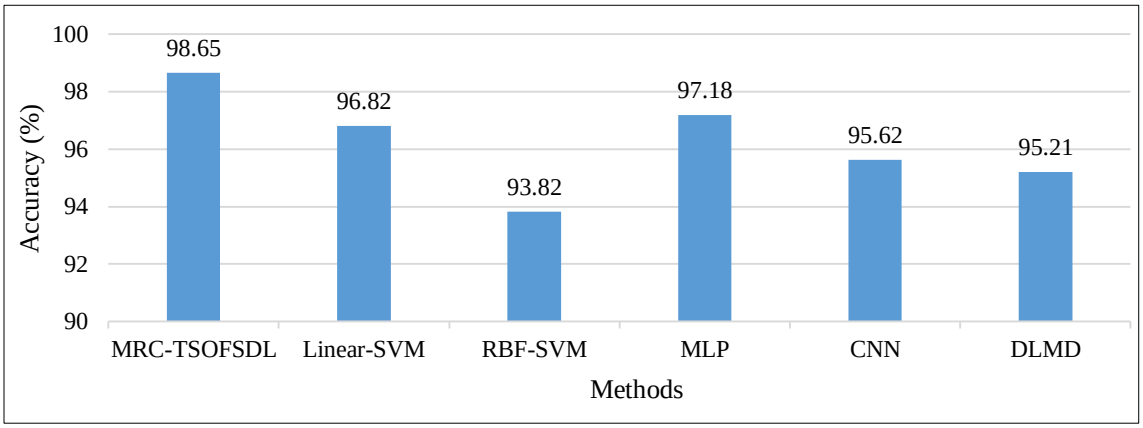

Fig. 10 Accuracy result of the MRC-TSOFSDL model with other techniques

The Computational Time (CT) results of the MRC-TSOFSDL technique with recent approaches are shown in Figure 11. These findings specify that the MLP and DLMD models showed increased CT values of 18.2s and 18.42s, respectively. However, RBF-SVM, linear SVM, and CNN models achieve closer CT values of 12.69s, 14.23s, and 9.62s; the MRC-TSOFSDL method is excellent than the other models with lessened CT of 5.09s. Thus, the MRC-TSOFSDL technique can be used for accurate malware detection.

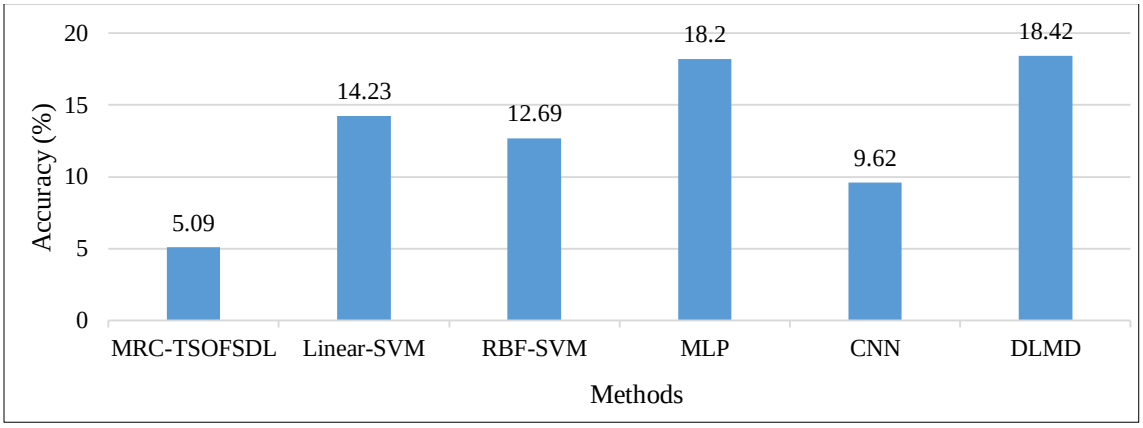

Fig. 11 CT outcome of MRC-TSOFSDL model with other techniques

5. Conclusion

In this research, a novel MRC-TSOFSDL technique is introduced. The MRC-TSOFSDL technique involves the concept of FS with an optimum hyperparameter selection strategy for enhancing the recognition results of the malware recognition process. It contains three significant processes involving TSO-based FS, SSAE-based classification, and ChoA-based parameter tuning. Initially, the MRC-TSOFSDL technique undergoes the TSO model and carries out the feature subset selection process. The SSAE model is followed for the automatic recognition and identification of malware. The ChoA-based hyperparameter tuning process is applied to improve the SSAE model's malware detection results. The performance analysis of the MRC-TSOFSDL technique is investigated using a malware dataset. The comparative results of the MRC-TSOFSDL technique demonstrated a maximum accuracy value of 98.65% over existing models. The limitations of the MRC-TSOFSDL technique comprise its reliance on specific datasets, which may hinder generalization, and the computational complexity that could affect scalability. Future work may concentrate on expanding the dataset for greater diversity, enhancing the algorithm's efficiency, exploring hybrid detection methods, and incorporating real-time detection capabilities to improve practical applicability.

References

[1] Abdulrahman Al-Abassi et al., "An Ensemble Deep Learning-Based Cyber-Attack Detection in Industrial Control System," *IEEE Access*, vol. 8, pp. 83965-83973, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Iqbal H. Sarker et al., "Intrudtree: A Machine Learning Based Cyber Security Intrusion Detection Model," *Symmetry*, vol. 12, no. 5, pp. 1-15, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Yihao Wan, and Tomislav Dragičević, "Data-Driven Cyber-Attack Detection of Intelligent Attacks in Islanded DC Microgrids," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 4, pp. 4293-4299, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Fargana J. Abdullayeva, "Detection of Cyberattacks in Cloud Computing Service Delivery Models Using Correlation Based Feature Selection," *IEEE 15th International Conference on Application of Information and Communication Technologies*, Baku, Azerbaijan, pp. 1-4, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Carmelo Ardito et al., "An Artificial Intelligence Cyberattack Detection System to Improve Threat Reaction in e-Health," *Italian Conference on Cybersecurity*, pp. 270-283, 2021. [Google Scholar] [Publisher Link]

[6] Md Mamunur Rashid et al., "Cyberattacks Detection in IoT-Based Smart City Applications Using Machine Learning Techniques," *International Journal of Environmental Research and Public Health*, vol. 17, no. 24, pp. 1-21, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Rahul Chourasiya, Vaibhav Patel, and Anurag Shrivastava, "Classification of Cyber Attack Using Machine Learning Technique at Microsoft Azure Cloud," *International Research Journal of Engineering & Applied Sciences*, pp. 4-8, 2018. [Google Scholar] [Publisher Link]

[8] Prabhat Kumar, Govind P. Gupta, and Rakesh Tripathi, "An Ensemble Learning and Fog-Cloud Architecture-Driven Cyber-Attack Detection Framework for IoMT Networks," *Computer Communications*, vol. 166, pp. 110-124, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Khoi Khac Nguyen et al., "Cyberattack Detection in Mobile Cloud Computing: A Deep Learning Approach," *IEEE Wireless Communications and Networking Conference*, Barcelona, Spain, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Xiayang Chen et al., "Ensemble Learning Methods for Power System Cyber-Attack Detection," *IEEE 3rd International Conference on Cloud Computing and Big Data Analysis*, Chengdu, China, pp. 613-616, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Subba Reddy Borra et al., "OEC-NET: Optimal Feature Selection-Based Email Classification Network Using Unsupervised Learning with Deep CNN Model," *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, vol. 7, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Muhammad Ajmal Azad et al., "DEEPSEL: A Novel Feature Selection for Early Identification of Malware in Mobile Applications," *Future Generation Computer Systems*, vol. 129, pp. 54-63, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Esraa Saleh Alomari et al., "Malware Detection Using Deep Learning and Correlation-Based Feature Selection," *Symmetry*, vol. 15, no. 1, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Yunchun Zhang et al., "Deep Hashing for Malware Family Classification and New Malware Identification," *IEEE Internet of Things Journal*, vol. 11, no. 16, pp. 26837-26851, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[15] Muhammad Shoaib Akhtar, and Tao Feng, "Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time," *Symmetry*, vol. 14, no. 11, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ragini Mokkapati, and Venkata Lakshmi Dasari, "Embedded Signal Artificial Neural Network Based Intelligent Non-Dependent Feature Selection for Cyber Attack Classification in Signal-Based Networks," *Signal Processing*, vol. 40, no. 3, pp. 905-914, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shamimul Qamar, "Healthcare Data Analysis by Feature Extraction and Classification Using Deep Learning with Cloud Based Cyber Security," *Computers and Electrical Engineering*, vol. 104, no. A, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Hashida Haidros Rahima Manzil, and S. Manohar Naik, "Android Ransomware Detection Using a Novel Hamming Distance Based Feature Selection," *Journal of Computer Virology and Hacking Techniques*, vol. 20, no. 1, pp. 71-93, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] G. Tirumala Vasu et al., "Improved Chimp Optimization Algorithm (ICOA) Feature Selection and Deep Neural Network Framework for Internet of Things (IOT) Based Android Malware Detection," *Measurement: Sensors*, vol. 28, pp. 1-8, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Nikolaos Polatidis et al., "FSSDroid: Feature Subset Selection for Android Malware Detection," *World Wide Web*, vol. 27, no. 5, pp. 1-17, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] Rajasekhar Chaganti, Vinayakumar Ravi, and Tuan D. Pham, "A Multi-View Feature Fusion Approach for Effective Malware Classification using Deep Learning," *Journal of Information Security and Applications*, vol. 72, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Akhil Mittal, and Pandi Kirupa Gopalakrishna Pandian, "Deep Learning Approaches to Malware Detection and Classification," *International Journal of Multidisciplinary Innovation and Research Methodology*, vol. 3, no. 1, pp. 70-76, 2024. [Google Scholar] [Publisher Link]

[23] Vinayakumar Ravi, and Mamoun Alazab, "Attention-Based Convolutional Neural Network Deep Learning Approach for Robust Malware Classification," *Computational Intelligence*, vol. 39, no. 1, pp. 145-168, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Uday Chandra Akuthota, and Lava Bhargava, "A Deep Learning Approach for the Detection of Intrusions with an Ensemble Feature Selection Method," *SN Computer Science*, vol. 5, no. 7, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Kamran Shaukat, Suhuai Luo, and Vijay Varadharajan, "A Novel Deep Learning-Based Approach for Malware Detection," *Engineering Applications of Artificial Intelligence*, vol. 122, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Sanjeev Kumar, B. Janet, and Subramanian Neelakantan, "IMCNN: Intelligent Malware Classification using Deep Convolution Neural Networks as Transfer Learning and Ensemble Learning in Honeypot Enabled Organizational Network," *Computer Communications*, vol. 216, pp. 16-33, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] S. Jebin Bose, and R. Kalaiselvi, "An Optimal Deep Learning-Based Framework for the Detection and Classification of Android Malware," *Journal of Intelligent & Fuzzy Systems*, vol. 44, no. 6, pp. 9297-9310, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[28] Changkang Sun et al., "An Enhanced FCM Clustering Method Based on Multi-Strategy Tuna Swarm Optimization," *Mathematics*, vol. 12, no. 3, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[29] Mhamad Bakro et al., "Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms along with Random Forest Model," *IEEE Access*, vol. 12, pp. 8846-8874, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[30] Chang'an Zhou et al., "Tool Wear Monitoring for Robotic Milling Based on Multi-Dimensional Stacked Sparse Autoencoders and Bidirectional LSTM Networks with Singularity Features," *Research Square*, pp. 1-22, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Fatma Refaat Ahmed et al., "A Novel Approach to Optimize LSTM Hyperparameter Using the Chimp Optimization Algorithm for the Pressure Ventilator Prediction," *Research Square*, pp. 1-27, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[32] Mansour Ahmadi et al., "Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification," *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, New York, United States, pp. 183-194, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[33] Muhammad Furqan Rafique et al., "Malware Classification Using Deep Learning Based Feature Extraction and Wrapper Based Feature Selection Technique," *Arxiv*, pp. 1-21, 2019. [CrossRef] [Google Scholar] [Publisher Link]