Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Mutation Dwarf Mongoose Optimization (Mdmo) Based Feature Engineering And Peephole Long Short-Term Memory (Plstm) Model For Malware Family Classification With Api Call Analysis

E. Yuvarani[1*], Dr. P. M. Gomathi[2]

1*Research Scholar, Department of Computer Science, P.K.R Arts College for Women, Gobichettipalayam, Erode, Tamil Nadu, Indiahttps://orcid.org/0009-0006-0949-8388, Email: yuvaranimca@gmail.com

[2]Associate Professor and Head, Department of Computer Science, P.K.R Arts College for Women, Gobichettipalayam, Erode, Tamil Nadu, India, https://orcid.org/0000-0002-8576-7512, Email: gomathipm@pkrarts.org

Abstract

Recently, there has been a notable surge in the development of malware. A serious security concern for both consumers and enterprises is malware development. The constant attempts by security research to prevent malware attacks are constantly prevented by malware developers. The best option for characterising malware behaviour is the Application Programming Interface (API). Achieving high classification accuracy (ACC) while concurrently resolving the problem of high-dimensional feature sets is the key obstacle when creating API call features for classification algorithms, particularly in fields like malware detection (MD). A new feature selection (FS) and classification model based on the Windows (MDS) malware detection system's API call sequence is presented in this research.Mutation (DMO) Dwarf Mongoose Optimization (MDMO) feature selection is introduced for selecting optimal features from API calls. The restricted way of prey capture (feeding) and foraging behaviour of the dwarf mongoose (DM) are mimicked by MDMO. In order to compensate for the effective feature selection from the API calls dataset, MDMO has significantly changed the social behaviour of the mongooses. To choose the best traits for malware classification, MDMO uses the dwarf mongoose's social groups such as the scout group, alpha group, and babysitters. To acquire integrated embedding from heterogeneous static and dynamic data, the MDMO technique is suggested. The PLSTM classifier is introduced for the purpose of determining hidden trends that many malware API calls generate. With the exception of the calculations for the peephole connections that the cell needs in order to control the gates, PLSTM works similarly to those of standard LSTM.The suggested approach is evaluated with the Windows MD Dataset on Kaggle. For simulation, the following performance metrics can be used: accuracy (ACC), precision (P), recall (R), and F1-score. From the outcomes, it is clear that the PLSTM can effectively boost the performance when compared to other methods using those metrics.

Keywords: Application Programming Interface (API), Mutation Dwarf Mongoose Optimization (MDMO), Peephole (LSTM) long short-term memory (PLSTM), feature engineering, malware detection (MD), classification, deep learning (DL), optimization.

1. Introduction

A software named malware that adversaries purposefully construct to harm or interfere with a computer system or network. The rise of the Internet of Things (IoT) and widespread internet connectivity have led to amalware attacks that are growing exponentially. Researchers require a sophisticated tool that can identify zero-day malware since the dynamics of malware are constantly evolving. One of the biggest threats to security professionals is malware, which presents a constant problem. Gathering a variety of data attributes, creating profiles, applying effective algorithms, and creating optimised threat detection and mitigation methods are all part of cyber threat intelligence (CTI) strategies. Either static attribute analysis or dynamic source parameter analysis can be used for MD. Now, the security experts' primary objective is to build effective and significant MD approaches for Windows-based malware [1].

In MD, the most effective method is Machine-learning (ML), in contrast to conventional detection techniques. The ML-based techniques have 2 main types. They are: Static analysis (SA)-based techniques and dynamic analysis (DA)-based techniques. Without actually running the sample, SA-based techniques examine its source code [2–3]. These techniques examine static components along all of a sample's execution routes, those components include strings, byte sequences, and operation codes (opcodes). This algorithm become ineffectual and susceptible to new malware versions, because zero-day, obfuscated, packed, polymorphic, or metamorphic malware attacks are not effectively detected by this algorithm [4-5].

Researchers have suggested DA-based techniques to address the aforementioned shortcomings [6-7]. The DA-based methods record a sample's behaviour in real time and the way other computer systems and the operating system (OS) communicate with it, they are resistant to the most recent malware. The DA-based methods run malware in a secure environment and record its behavioural traits through registry and memory changes, network interactions, and API calls, it provides real intention of the malware, unlike SA. Researchers can successfully categorise malware by examining the features in the reports, which are created throughout the tracking procedure and record the procedures that the software performs (such as file access, API calls, network connectivity, etc.). Several types of malware are depicted in Figure 1 [8].

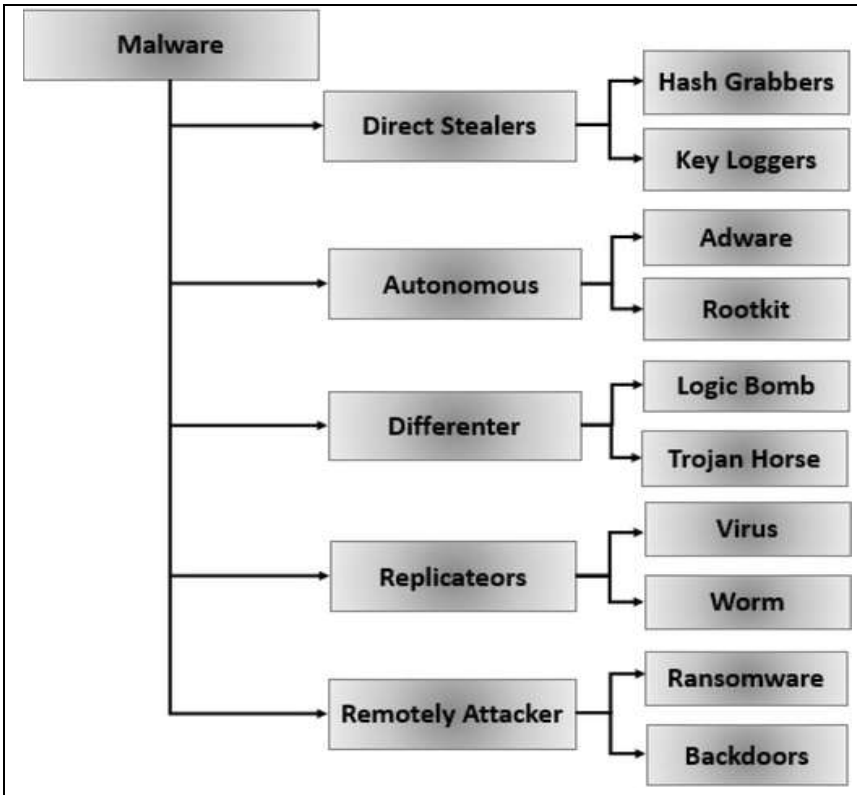

Figure 1: Malware Classes [8]

An increase in malware attacks cannot be prevented by previous methods. This prompted a number of researchers to suggest several malware classification schemes. Malware classification is challenging, especially due to the limitations of relying solely on antivirus programs. Analyzing API call sequences and their shared characteristics can offer a more reliable way to identify malware families, as similar families tend to exhibit consistent behavior patterns. Research on classifying malicious programs into families is lacking. Malware feature generation and feature learning are the two phases of malware family classification based on assembly instruction and visualisation technology.

There are a number of significant issues with the majority of current machine learning techniques, including: (1) Identifying effective and precise features to classify malware; (2) necessitating extensive feature engineering or highly specialised domain expertise to create the features; (3) a low classification ACC rate; and

(4) an imbalance in the data that makes malware classification and accurate identification challenging. However, manual design and FS are required by traditional ML approaches. Therefore, these traditional ML techniques may not be able to cover all malware characteristics and can be time-consuming. [9]. In large datasets, complex patterns can be effectively detected via the Deep learning (DL) method.To enhance the effectiveness of MD, DL manages massive amounts of data and automatically extracts useful features. In reality, it is difficult to get malware samples, and for some sophisticated viruses, the sample size is also limited to train models effectively. The classification performance and generalisation capabilities of DL techniques are severely hampered by this constraint, which causes overfitting problems [10]. To improve classification performance, a greater number of features should be chosen.

By verifying various attribute combinations in a dataset, FS algorithms can minimise the amount of features needed to construct an ML model. The optimization techniques are utilized by the FS. This utilization may support in determining an optimal feature subset from a dataset. Thus, model's efficiency was enhanced by this application, and it also lessens the feature count that are used [11–12]. The most distinctiveattributes are now frequently detected using meta-heuristic optimisation algorithms (MHOA) [12–13]. For FS, DMO is a novel method that imitates the dwarf mongoose's foraging behaviour [14]. Recurrent (NN) neural networks (RNNs) of the LSTM classifier type are ideal for classification since they are specifically made to handle sequential data with long-range dependencies.

Based on the Windows MDS API call sequence, a novel FS and classification model is suggested in this research. To choose the best features, MDMO FS is presented. To choose the best traits for malware classification, MDMO uses the dwarf mongoose's social groups, such as the scout group, alpha group, and babysitters. The PLSTM classifier is introduced for the purpose of determining hidden trends that many malware API calls generate. Compared to alternative approaches, PLSTM can significantly improve performance.

2. Literature Review

A malware family classification (MFC) method that represents each malware instance as a Behavioral Tree (BT) has been suggested by Xu and Chen [14]. The API call sequence obtained from dynamic malware research is used to build this BT. Its ability to depict the control structure and connections among the API calls is described. Record a set of binary relations from the BT known as Heighted Behavior Relations as malware behavior features to lower the computational complexity. Term Frequency-inverse Document Frequency (TF-IDF) technology is used, as it converts malware behavioural features into family behavioural features. The similarity between each malware and all the families is then used to build its similarity vector. The Naïve Bayes (NB) algorithm is effective for malware classification. With the help of NB, a classifier is trained using malware similarity vectors, and it may contribute in family classification. For simulation purpose, dataset with 10620 malware samples from 43 malware families are used. Based on API call sequences, suggested approach's classification accuracy (ACC) is 10% greater than that of the traditional approaches.

The visualisation analysis and Jaccard similarity (JS) was used by Daeef et al. [15]. This suggestion may support in detecting hidden patterns that several malware API calls may generate. Traditional ML classifiers like Random Forest (RF), Support Vector Machine (SVM), and K Nearest Neighbour (KNN) are utilised as alternatives to the present NN, which uses API call sequences of millions of lengths.Then, 7107 samples of API call sequences are contained in the benchmark dataset, and it is used in this study.

For interpretable and large-scale classification, a unique MFC approach (MalAtt) was suggested by Bao et al. [16]. From the heterogeneous static and dynamic data, anintegrated embedding is obtained by using a unified feature engineering technique. Semantic features are then gradually extracted from the static and dynamic embedding using a hierarchical attention encoder module. To represent the relationships and interactions between high-level static features and dynamic features, a Collaborative Attention Module (CAM) is presented. In order to implement MFC, MalAtt can be plugged for many classifiers, including SVM and RF. For analysis, a dataset including over 20,070 real-world samples from 62 malware families is collected.

An automated FS technique that uses Particle Swarm Optimisation (PSO) for behavior-based (RD) ransomware detection and classification was introduced by Abbasi et al. [17]. The suggested approach executes FS based on

the relevance of the feature groups, considering the significance of each group in RD and identification. According to the simulation outcomes, the suggested approach overtakes previous state-of-the-art (SOTA) benchmarking techniques by an equivalent amount or a large margin. This method more successfully detects ransomware behavior patterns by analysing the significance of complete feature groups, and selects attributes.

MalAnalyser is a new and lightweight Windows MDS based on API call sequences that was suggested by Dabas and Sharma [18]. It initially extracts frequent patterns from API call sequences in order to filter unnecessary API calls and highlight crucial information regarding malware samples.After that, it finds a limited number of important features that contribute to MD by using the Global Local Best PSO (GLBPSO) algorithm to common patterns. The frequent malware patterns are enhanced by the MalAnalyser with the support of Genetic Algorithm (GA). This will facilitate in detecting unseen malware behavior.To assess how well MalAnalyser performed on various datasets and settings, lots of experimentation was done. A benchmark dataset was also used to assess MalAnalyser's performance; the results showed that MalAnalyser outperforms similar techniques.

Swarm Optimisation (SO) and ML Applied to PE MD (SOMLAP) is a new dataset that Kattamuri et al. [19] created in addition to the pre-existing benchmark dataset. The SOMLAP data contains 51,409 samples of malware and benign files with 108 pure PE file header attributes. Using SO tools, namely Ant Colony Optimisation (ACO), Cuckoo Search Optimisation (CSO), and Grey Wolf Optimisation (GWO) in conjunction with classification techniques (KNN, GNB, SVM, RF, and DT), features were minimised forimproving the MDS efficacy.

In order to capture and combine additional significant elements, known as intrinsic features of the API sequence, Li et al. [20] developed a unique MDS employing DL models. In order to capture the software's behaviour, convolutional layers (CL) and embedding are specifically introduced to perform a joint representation of several APIs. Second, each API call's semantic information is represented by the category, action, and operation object of the API. In order to find the relationship data between APIs, the Bidirectional LSTM (Bi-LSTM) classifier is finally introduced.

In order to classify malware variants utilising long-sequences of API calls, Li and Zheng [21] introduced the LSTM and the gated recurrent unit (GRU). To increase the classification ACC and reduce noise in lengthy API call sequences, a pre-processing approach is suggested. To make the sequence suit the suggested LSTM model, shorten it and eliminate the same API from it repeatedly. Gradients of errors are frequently propagated via Backpropagation (BP)-Through-Time (BPTT). A variation of the BP technique for feedforward NN (FFNN) training is called BPTT. The optimisation technique Adagrad uses a per-parameter learning rate and is a modified stochastic gradient descent method (SGD). The classification techniques are validated using P, R, F1-score, and ACC.

By considering the behaviour of malware, Catak et al. [22] created a classification approach based on malware kinds. A new dataset that includes Windows (OS) API calls that illustrate the activities of suspiciousprogram has been gathered. The dataset contains the following categories of malicious malware: Adware, Backdoor, Downloader, Dropper, spyware, Trojan, Virus, and Worm. One popular classification technique for sequential data is LSTM. Binary class (BC) and multi-class (MC) malware datasets are used to simulate the outcomes and assess how well the LSTM model performs.

An investigation into the distinctions and relationships between malicious programs' static and dynamic API sequences was suggested by Han et al. [23]. A hybrid feature vector space is created by combining the static and dynamic API sequences into a single, cohesive hybrid sequence using semantic mapping (SM). Additionally, mining the programs, identify the forms of malicious behaviour, and give MD results that can be explained. Create an explainable MDS known as MalDAE by fusing and correlating the static and dynamic API sequences. MalDAE gives analytical support for comprehending and preventing malware, along with a comprehensive overview of common malware kinds.

The semantic and sequence features of the API calls are integrated in a novel MDS, and it has been suggested by Zhang et al. [24]. API functions or sequences can be represented in low dimensions using the API2Vec embedding technique. To find the sequential segment’s behavioural features, the features are extracted by the

balts. Utilising the implicit semantic data of the API functions requires retrieving the operation and resource type. The attention-related modules subsequently combine and process these sequential and semantic data. When it comes to semantic modelling and identifying crucial sequences in API call sequences, Mal-ASSF outperforms the current approaches.

3. Proposed Methodology

In this paper, a novel FS and classification model is proposed based on API call sequence in Windows MDS. MDMOFS is introduced for selecting optimal features from API calls. The restricted way of prey capture (feeding) and foraging behaviour of the dwarf mongoose are mimicked by MDMO. In order to compensate for the effective FS from the API calls dataset, MDMO has significantly changed the social behaviour of the mongooses. To choose the best traits for malware classification, MDMO uses the dwarf mongoose's social groups such as the scout group, alpha group, and babysitters. To acquire integrated embedding from heterogeneous static and dynamic data, the MDMO technique is suggested.PLSTM classifier is introduced for the purpose of determining hidden trends that many malware API calls generate. With the exception of the calculations for the peephole connections that the cell needs in order to control the gates, PLSTM works similarly to those of standard LSTM. The suggested approach is evaluated with the Windows MD Dataset on Kaggle. PLSTM can effectively boost the performance when compared to other methods using P, R, F1-score, and ACC.

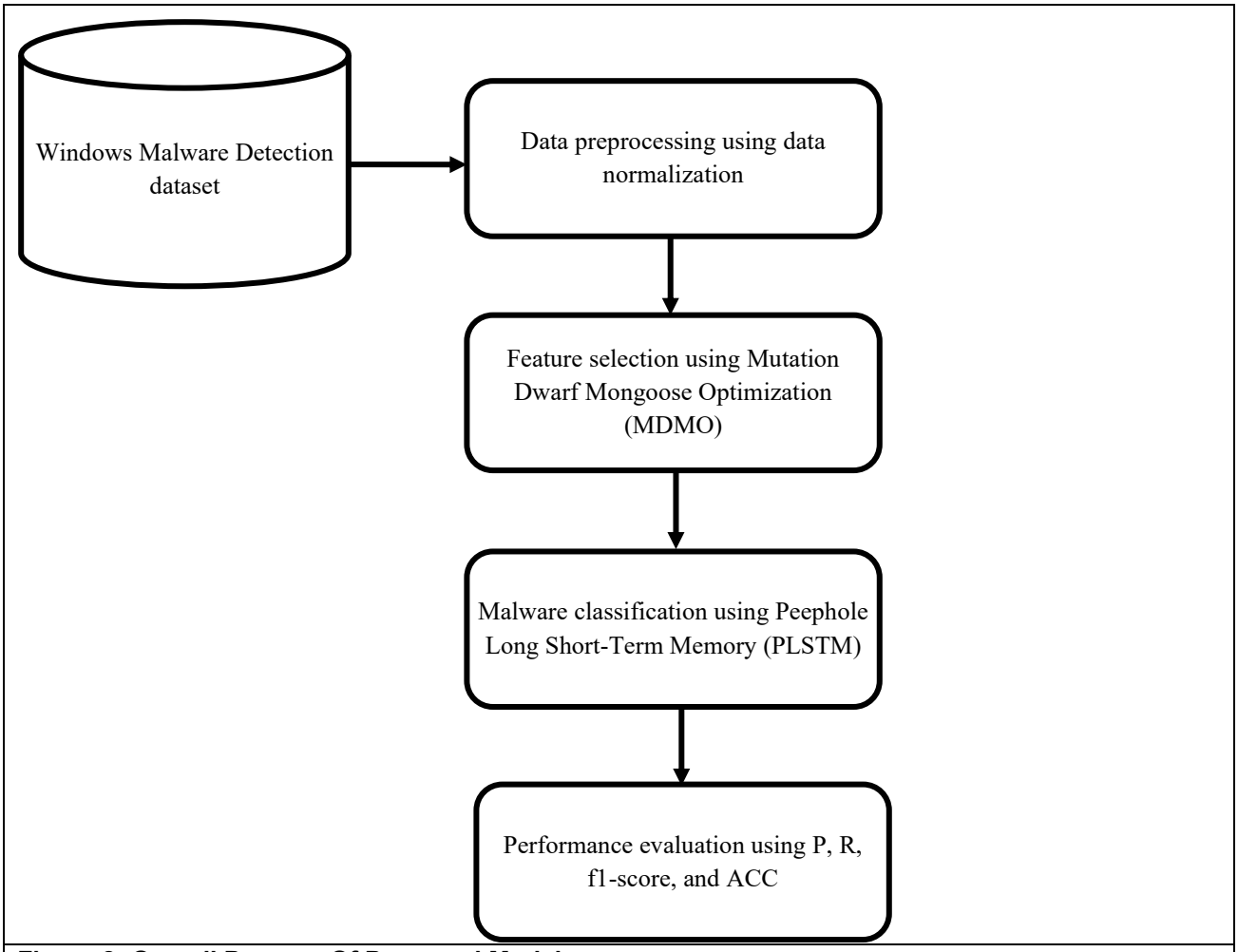

Figure 2: Overall Process Of Proposed Model

3.1. Dataset

There are four feature sets in Windows Portable Executable Samples. 4 CSV files total, one for each feature set are included.

1. The 1st feature set (DLLs_Imported.csv file) contains a list of the imported DLLs for each malware family. The 1st column lists Secure Hash Algorithm 256 (SHA256). The 2nd is the malware's family type or labelling. The remaining columns include the names of imported DLL.
2. 2nd feature set (API_Functions.csv files) includes the SHA256 hash values and labels of the API functions that the malware calls.
3. The values of 52 PE header fields are contained in the 3rd feature set (PE_Header.csv). The CSV file has labels for every field.
4. 9 field values from 10 distinct PE sections are included in the 4th feature set (PE_Section.csv file). The CSV file has labels for every field.

Only 2 and 3 are taken into consideration for doing experiments.

3.2. Data Normalization (Dn)

One method for improving data integrity and minimising redundancy is DN. To ensure consistency and remove data duplication, it basically entails organising data into tables and specifying relationships between them. This process is crucial for efficient database management, making it easier to update, query, and maintain the data while minimizing storage space. One DN method that converts data to have a mean of 0 and a SD of 1 is called Z-score (ZS) normalisation (ZSN). This approach adjusts the data according to the number of standard deviations (SD) that each data point (DP) deviates from the mean, centring it around 0. Here is the ZSN equation (1) [25].

$$Z = (x - \mu) / \sigma \quad (1)$$

Here, an initial data point is denoted by x. The mean of the data is represented by μ. The SD is indicated by σ.

3.3. Mutation Dwarf Mongoose Optimization (Mdmo) Based Feature Selection

An animal-inspired swarm intelligence (SI)-based technique named DMO is used. Using Portable Executable (PE) samples, this DMO determines the global optimal features. It mimics the behavioural behaviours of dwarf mongooses (DM). When it comes to solving difficult FS problems, DMO performs admirably in MFC FS.The modest convergence rate and near-perfect stagnation of DMO, still require improvement. To get around these issues with neighbouring local optima and enhance global search (GS) capabilities, MDMO and ZS distributions were introduced. The DM's randomly created population is the first step. To choose the best features from malware samples, the DM population in the DMO uses three social categories: scouts, babysitters, and the alpha group. The family's hunting activities are led by the alpha femaleand finds the best features from malware samples by identifying the sleeping mounds, distance travelled, and foraging path.A subset of DM population that contains male and female types offers babysitters for selecting optimal attributes from malware samples. For the best features from malware samples, stay behind with the children till the remainder of the group arrives in the noon. To start foraging with the group for the best features from malware samples, the babysitters are first switched [26]. In order to find a fresh foraged location for parameter tuning, the DM group constantly moves the sleeping mound rather than building a nesting site to shelter the young. In a region big enough to support an exploration phase for the best FS from malware samples, the DMs have developed a seminomadic way of life [27]. The following compensatory behavioural response of the DM is replicated by DMO [28].

Alpha Group: After initializing population, the efficiency of optimal features is computed. The probability value is obtained by equation (2), and it is used for choosing α.

$$\alpha = \frac{fit_i}{\sum_{i=1}^{n_{bs}} fit_i} \quad (2)$$

The mongoose count in the α is related to the n_{bs}. The babysitters count is denoted bybs. The dominant female's vocalisation, known as peep, keeps the family on course for the best FS [28]. The following is the mechanism for updating the tune hyperparameters solutions:

$$X_{i+1} = X_i + ph_i * peep \tag{3}$$

For every iteration, a distributed random number within an interval [1,1] is denoted as ph_i. The sleeping mound is shown in equation (4).

$$si_m = \frac{fit_{i+1} - fit_i}{\max[|fit_{i+1} - fit_i|]} \tag{4}$$

The average number of sleeping mounds found is provided in equation (5).

$$\varphi = \frac{\sum_{i=1}^{n} si_m}{n} \tag{5}$$

The procedureproceeds the scouting stage after the babysitting exchange condition is met. In this case, the next food source or resting mound determines the ideal FS.

Scout Group: The family will find an ideal sleeping mound for the best FS if they forage widely in the scout group area. Equation (6) simulates the scout mongoose.

$$X_{i+1} = \begin{cases} X_i - CF * ph_i * rand\left[X_i - \vec{M}\right], if\varphi_{i+1} > \varphi_i \\ X_i + CF * ph_i * rand\left[X_i - \vec{M}\right], \quad else \end{cases} \tag{6}$$

Here, the arbitraryvaluein [0,1] is denoted as $rand$.

$$CF = \left(1 - \frac{iter}{Max_{iter}}\right)^{\left(2* \frac{iter}{Max_{iter}}\right)} \tag{7}$$

$$\vec{M} = \sum_{i=1}^{n} \frac{X_i * sm_i}{X_i} \tag{8}$$

The lower-ranking group members serve as babysitters for the young, the alpha female, or mother, is in charge of hunting expeditions.

Babysitters Group

The lower-ranking group members serve as babysitters for the young, the alpha female, or mother, is in charge of hunting expeditions. The relationship among the babysitters count and population size are directly proportional. Then, the food source and scouting data of the family are reset by exchange parameters of the babysitter. Here, the value 0 is assigned as fitness weight of the babysitter, and this will support in preventing group migration and rising exploitation. It also ensures that the alpha group's average weight is decreased in the following iteration.

To improve feature search space exploration and avoid local optima, the ZS distribution is introduced. By randomly perturbing a solution using a ZS distribution, it diversity is introduced which helps to find optimal features in the malware family classification. This approach is particularly effective when the search space has multiple local optima, as it can guide the algorithm towards the global optimum. With a mean of 0 and a SD of 1, ZS is a normal distribution. Mutation is expressed in the following manner.

$$mutant(x) = x_i * 1 + Z \tag{9}$$

Here, Z is the random value drawn from the ZS distribution using equation (1), and x_i is the parameter's current value. ZS distribution is the mutation operation that population variability and explores the search space. The ZS distribution emphasizes a local search close to the original individual best features based on the normal distribution and it has demonstrated excellent efficiency. Every feature is then chosen, and the equation (10), is used to evaluate it. It is interpreted as malware or non-malware in order to reduce the classifier error.

$$fitness(x) = Minimum(Error\ Rate) \tag{10}$$

$$Error\ Rate = \frac{Number\ of\ misclassified\ malwares}{Total\ number\ of\ samples} * 100 \quad (11)$$

Algorithm 1 describes optimal FS in MDMO.

PROCEDURE 1. PSEUDOCODE OF THE MDMO
1. Initialize the algorithm with malware features and solutions
2. While ($iter < Max_{iter}$) do
3. for ($i = 1\ to$ n) do
4. Compute the Fitness Function (FF) of Mongoose by equations (10-11).
5. Alpha value is computed by using Equation (2).
6. Hyperparameter solution is found by using Equation (3).
7. Using Equation (4) to evaluate the sleeping mound.
8. By using Equation (5), an average of the sleeping mound is computed.
9. Movement vector is computed by using Equation (7).
10. Use Equation (4) to model the scout Mongoose for the subsequent hyperparameter solution.
11. end for
12. t = t + 1
13. end while
14. Return the optimal features solution (x)

3.4. Plstm Based Malware Classification

For MFC, the PLSTM model is presented. In order to control the gates for MFC, the cell needs a peephole connection, which PLSTM has as an option. The performance of the network is impacted by a loss of crucial information since the LSTM cells' gates lack direct connections from the cell state, which can only be accomplished by an open output gate. Additionally, gates can utilise the prior internal and hidden states due to peephole connections. In comparison to conventional LSTM cells, this enables PLSTM to learn more accurate timings [29]. With the exception of the computations for the peephole connections, the PLSTM operates identically to the conventional LSTM [30].

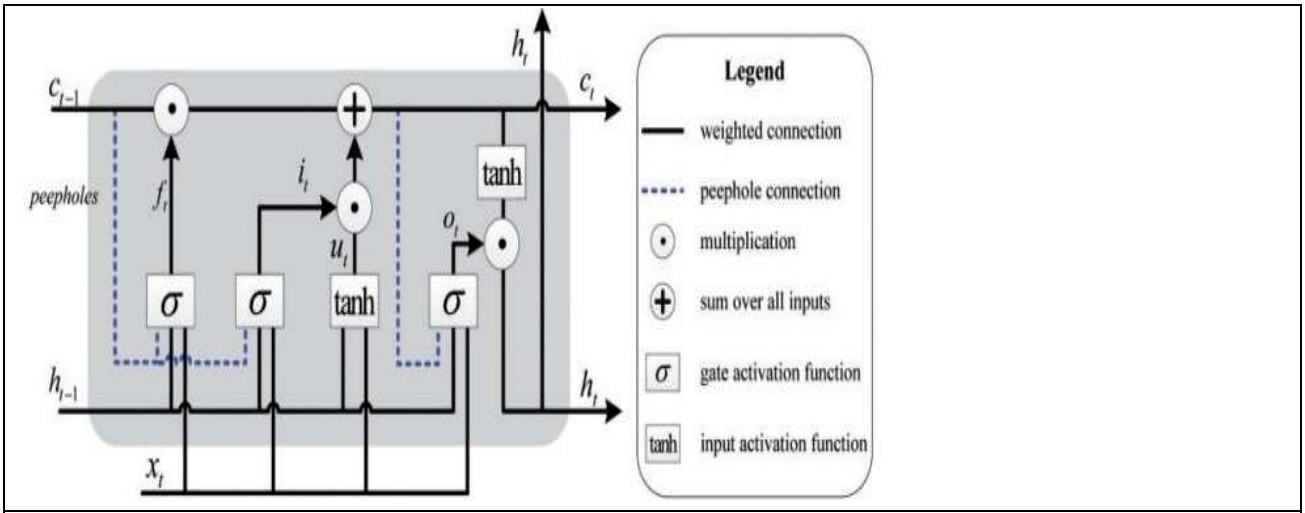

Figure 3. Plstm configuration

A PLSTM structure can be seen in Figure 3. The fundamental LSTM design has multiple variations [30]. By establishing peephole connections, the gates depends on the previous internal state c_{t-1}, input layer x_t and the prior hidden state h_{t-1}. The forget gate $f_t$is produced by the (AF) activation function δ_g, and a term that also comes from the cell c_t is added to the gate equations (12–17). The f_t output is evaluated using the sigmodal

output, which has a range of 0 to 1. The ability to forget to hide the cell's state in the top layer is demonstrated by this output.

The description of equation (12) is given below:

$$f_t = \delta_g\left(We_f x_t + V_f h_{t-1} + U_f c_{t-1} + bi_f\right) \tag{12}$$

The input gate processes the actual series position input. The cell state, input MD dataset, and peephole connection value are the three components of this input gate. Equation (13) indicates that the output is i_t when the sigmoid function is enabled.

$$i_t = \delta_g(We_i x_t + V_i h_{t-1} + U_i c_{t-1} + bi_i) \tag{13}$$

Then, two components are combined to update the cell state. The activation function's output of x_t, c_{t-1}, h_{t-1} creates the second portion, whereas c_{t-1} and the f_t's output are multiplied to create the first element. Equation (14) provides the following updated cell state:

$$c_t = i_t \delta_c\left(We_c x_t + U_f c_{t-1} + V_f h_{t-1} + bi_i\right) + c_{t-1} f_t \tag{14}$$

The input value, peephole value, and $c_t$all affect the output gate's decision. The new c_t is also processed by memory. The linked cell state and output gate value are determined by the sigmoid function in equation (15).

$$o_t = \delta_g(We_o x_t + U_o c_{t-1} + V_o h_{t-1} + bi_o) \tag{15}$$

Using equation (16), the o_t and the cell state that the function activates are multiplied to update the hidden state.

$$h_t = o_g \delta_h(c_t) \tag{16}$$

The output of the whole PLSTM model is expressed as:

$$y_t = k\left(We_h h_t + bi_y\right) \tag{17}$$

where the weight matrices are $We_f, We_i, We_o, V_f, V_i, V_o$, and U_f, U_i, U_o. The hidden output weight matrix is denoted by We_h. T he bias vectors are bi_f, bi_i, bi_o, bi_y. Along with producing reports that include DA and SA results, such as API calls, PE files, and hash values, this classifier keeps an eye on each malware's behaviour. Each sample's hash value is then supplied to the classifier, which uses various antivirus software to examine each sample. The majority agreement among antivirus engine results is used to establish the malware's class because several antivirus engines generate diverse labels for the similar sample. By comparing the malware classes, API call sequences, and PE file sequences, the labelled dataset was finally generated.

4. Experimental Results And Discussion

Simulation are conducted to assess both the suggested approach and current approaches. A running Windows 10 with an 11th Gen Intel(R) Core(TM) i5- 11600KF @ 3.90 GHz CPU, 32 GB of RAM, and a GeForce RTX 3060 Ti GPU was used to run and test the model. The Python 3.9.1 framework was developed using the TensorFlow 2.3.0 and Keras 2.7.0 frameworks. Public access to the Scikit-learn, Numpy, Pandas, Matplotlib, Seaborn, LIME, and Natural Language Toolkit (NLTK) modules is possible through the Python package management website PiPy (Pypi, 2021).MalAnalyser, MalDAE, RF, SVM, LSTM, GRU, and PLSTM are examples of traditional ML methods.

4.1. Dataset

The Windows MD dataset, which may be found at https://figshare.com/articles/dataset/Windows_Malware_Detection_Dataset/21608262, has malware type/family labels such as 0=Benign, 1=RedLineStealer, 2=Downloader, 3=RAT, 4=BankingTrojan, 5=SnakeKeyLogger, and 6=Spyware. A multi-classification problem on PE files on Windows systems serves as the basis for the findings. The suggested framework is based on the DA technique, which uses a virtual isolated environment to extract API call sequences from executable files, both malicious and benign.

4.2. Evaluation Metrics

P, R, F1-score, and ACC are four important metrics are utilized to assess the effectiveness of the recommendedframework. The amount of samples that the model properly identified and classified in a specific malware category is denoted by True Positive (TP). The sample count that are accurately identified as not falling within a certain malware category is known as True Negative (TN). The amount of times a model inaccurately classifies samples that do not correspond to a specific malware category as belonging to it, and it is called as the False Positive (FP). False Negative (FN) refers to the amount of times the model incorrectly classifies samples that fall within a specific malware category as not belonging to it.

$$P = \frac{TP}{TP + FP} \tag{18}$$

$$R = \frac{TP}{TP + FN} \tag{19}$$

$$F1 - Score = 2 \times \frac{P * R}{P * R} \tag{20}$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{21}$$

Table 1. P, R, F1-Score, And Acc On Datasets

FS methods	Classification methods	API_Functions (%)- D1			
		Precision	Recall	F1-Score	Accuracy
GLBPSO	MalAnalyser	90.29	86.07	88.13	86.07
GLBPSO	MalDAE	91.10	87.28	89.15	87.28
GWO	SVM	92.22	88.83	90.49	88.76
CSO	LSTM	92.97	89.82	91.37	89.81
GWO	GRU	94.01	91.24	92.61	91.27
MDMO	PLSTM	95.27	93.01	94.13	92.96
FS methods	Classification methods	Portable Executable (PE) files (%)-D2			
		Precision	Recall	F1-Score	Accuracy
GLBPSO	MalAnalyser	88.96	84.29	86.54	84.29
GLBPSO	MalDAE	89.90	85.57	87.69	85.58
GWO	SVM	91.00	87.12	89.00	87.02
CSO	LSTM	91.95	88.42	90.11	88.38
GWO	GRU	93.26	90.21	91.69	90.16
MDMO	PLSTM	94.25	91.61	92.88	91.54

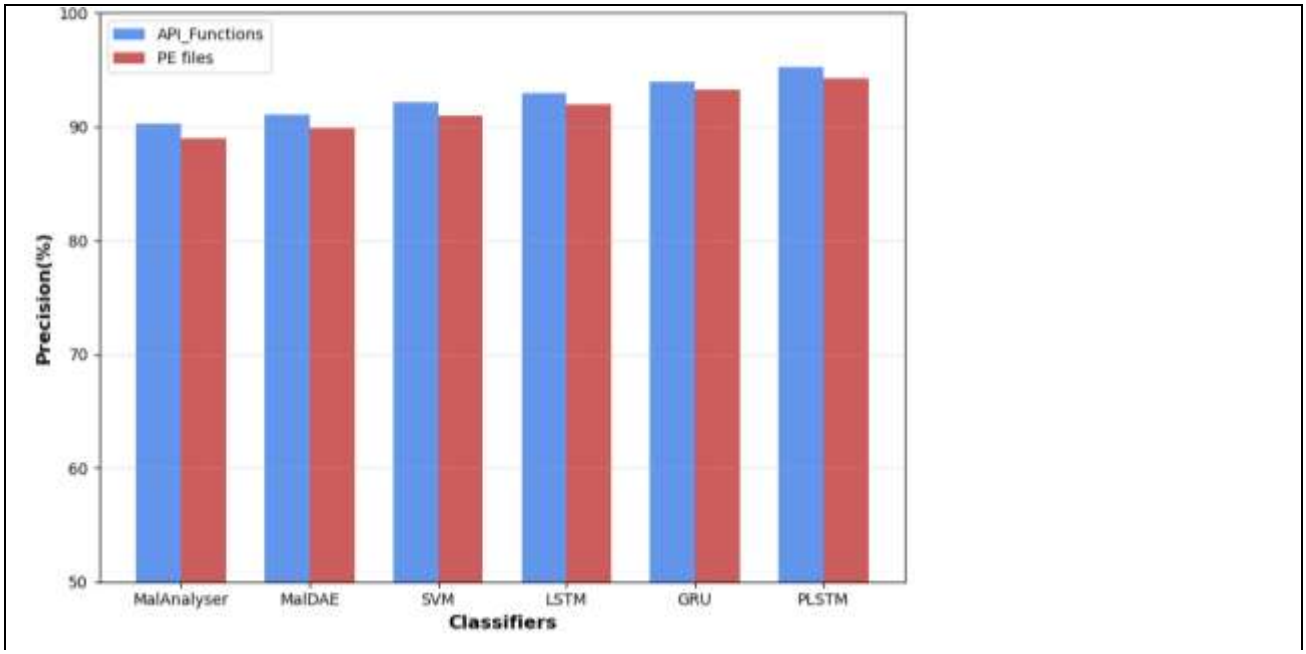

Figure 4: Precision Analysis Of Classification Methods

The comparative analysis of classification methods like MalAnalyser, MalDAE, SVM, LSTM, GRU, and PLSTM against two datasets (API_Functions, and PE files) in terms of P was visualized in Figure 4. The suggested classifier attainstopPoutcomes of 95.27%, and 94.25% for API_Functions, and PE files. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest precision of 90.29%, 91.10%, 92.22%, 92.97%, and 94.01% for API_Functions. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest precision of 88.96%, 89.90%, 91.00%, 91.95%, and 93.26% for PE files. Proposed system has highest results due to optimal selection of features from PE files, and API calls. MalAnalyser, MalDAE, SVM, LSTM, GRU, and PLSTM methods against two datasets (API_Functions, and PE files) with respect to recall are presented in figure 5. The suggested classifier attains highest recall results of 93.01%, and 91.61% for API_Functions, and PE files. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest recall of 86.07%, 87.28%, 88.83%, 89.82%, and 91.24% for API_Functions. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest recall of 84.29%, 85.57%, 87.12%, 88.42%, and 90.21% for PE files. High R value is attained by the suggested method when compared to current methods. When the malware samples in a dataset is higher than a realistic ratio, the suggested method that utilizes an executable's API call sequences detects malicious behavior more effectively.

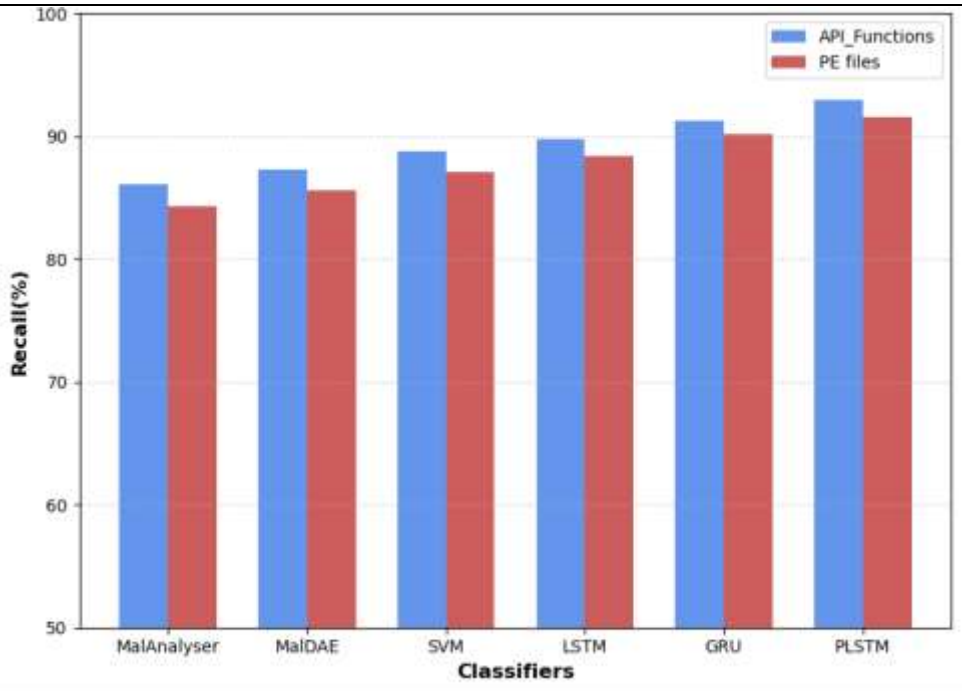

Figure 5: Recall Analysis Of Classification Methods

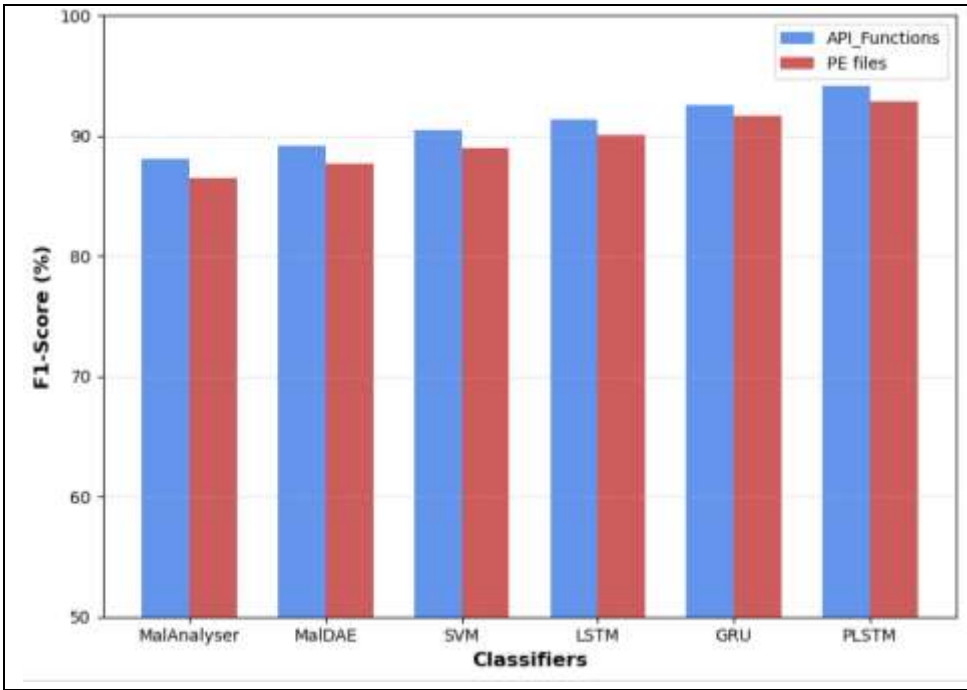

Figure 6: F1-Score Analysis Of Classification Methods

MalAnalyser, MalDAE, SVM, LSTM, GRU, and PLSTM methods against two datasets (API_Functions, and PE files) with respect to f1-score are presented in figure 6. The suggested classifier has highest f1-score outcomes of 94.13%, and 92.88% for API_Functions, and PE files. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest f1-score of 88.13%, 89.15%, 90.49%, 91.37%, and 92.61% for API_Functions. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest f1-score of 86.54%, 87.69%, 89.00%, 90.11%, and 91.69% for PE files. Then, the final results obtained across two datasets demonstrate that the suggested classifier may be able to identify both known and unknown malware attacks.

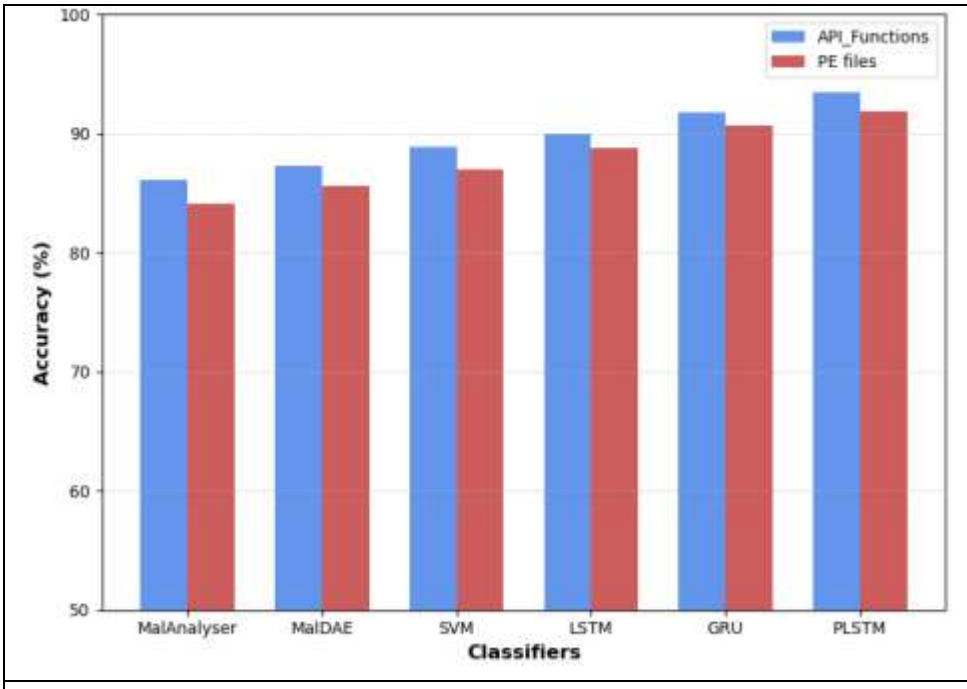

Figure 7: Accuracy Analysis Of Classification Methods

MalAnalyser, MalDAE, SVM, LSTM, GRU, and PLSTM methods against two datasets (API_Functions, and PE files) in terms of ACC are presented in figure 7. The suggested classifier attains highest ACCoutcomes of 92.96%, and 91.54% for API_Functions, and PE files. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest accuracy of 86.07%, 87.28%, 88.76%, 89.81%, and 91.27% for API_Functions. Other methods such as MalAnalyser, MalDAE, SVM, LSTM, GRU gives lowest accuracy of 84.29%, 85.58%, 87.02%, 88.38%, and 90.16% for PE files. Then, the suggested classifier may be able to identify both known and unknown malware attacks, as demonstrated by the overall results obtained across two datasets.

5. Conclusion And Future Work

To detect malware attacks on Windows systems, a novel automated framework based on DL is presented in this study.Initially, Windows Malware Detection Dataset is collected from Kaggle. Secondly, ZSN is introduced which transforms data with zero mean and a standard deviation of 1. Then, a small number of important features that aid in MD are found by applying the MDMO algorithm to samples. 3 social categories used by the DM population according to the MDMO algorithm. MDMO, ZS distribution with mutation is introduced to solve local optima problem, and improve global search ability. By randomly perturbing a solution using a ZS distribution, it diversity is introduced which helps to find optimal features in the malware family classification. Peephole connections are then incorporated into the PLSTM classifier, which enables the gate layers to directly access the cell state for classification. It enables the gates to make decisions based on the current memory cell state, potentially improving performance in malware detection and long-term dependency modeling.On various benchmark datasets, PLSTM performs better than current SOTA MD methods by P, R, F1-score, and ACC. In windows systems, the malware attack and benign activity are effectively separated by the suggested method as it detects distinct patterns from API call sequence, and it was demonstrated by the outcomes. Examine the suggested framework's ability to extract API call features from Android apps, here characteristics from APK files will be dynamically evaluated. Additionally, adversarial samples, which can deceive MD models based on API calls will be employedforassessing the suggested model.

References

1. Han, W., Xue, J., Wang, Y., Huang, L., Kong, Z. and Mao, L., 2019. MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. computers & security, 83, pp.208-233.
2. Ucci, D., Aniello, L. and Baldoni, R., 2019. Survey of machine learning techniques for malware analysis. Computers & Security, 81, pp.123-147.
3. Rizvi, S.K.J.; Aslam, W.; Shahzad, M.; Saleem, S.; Fraz, M. PROUD-MAL: Static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable. Complex Intell. Syst. 2022, 8, 673–685.
4. Johnson, S.; Gowtham, R.; Nair, A.R. Ensemble Model Ransomware Classification: A Static Analysis-based Approach. Inventive Comput. Inf. Technol. 2022, 336, 153–167.
5. Xiaofeng, L., Fangshuo, J., Xiao, Z., Shengwei, Y., Jing, S. and Lio, P., 2019. ASSCA: API sequence and statistics features combined architecture for malware detection. Computer Networks, 157, pp.99-111.
6. Daku, H., Zavarsky, P. and Malik, Y., 2018, Behavioral-based classification and identification of ransomware variants using machine learning. In 2018 17th IEEE international conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE), pp. 1560-1564.
7. Jing, C.; Wu, Y.; Cui, C. Ensemble dynamic behavior detection method for adversarial malware. Future Gener. Comput. Syst. 2022, 30, 193–206.
8. Poudyal, S.; Akhtar, Z.; Dasgupta, D.; Gupta, K.D. Malware analytics: Review of data mining, machine learning and big data perspectives. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 649–656.
9. Vasani, V., Bairwa, A.K., Joshi, S., Pljonkin, A., Kaur, M. and Amoon, M., 2023. Comprehensive analysis of advanced techniques and vital tools for detecting malware intrusion. Electronics, 12(20), p.4299.
10. Karnati M, Seal A, Bhattacharjee D et al (2023) Understanding deep learning techniques for recognition of human emotions using facial expressions: a comprehensive survey. IEEE Trans Instrum Meas 72:1–31.

11. Akinola, O.O., Ezugwu, A.E., Agushaka, J.O., Zitar, R.A. and Abualigah, L., 2022. Multiclass feature selection with metaheuristic optimization algorithms: a review. Neural Computing and Applications, 34(22), pp.19751-19790.
12. Kaur, S., Kumar, Y., Koul, A. and Kumar Kamboj, S., 2023. A systematic review on metaheuristic optimization techniques for feature selections in disease diagnosis: open issues and challenges. Archives of Computational Methods in Engineering, 30(3), pp.1863-1895.
13. Abdollahzadeh, B. and Gharehchopogh, F.S., 2022. A multi-objective optimization algorithm for feature selection problems. Engineering with Computers, 38(Suppl 3), pp.1845-1863.
14. Xu, Y. and Chen, Z., 2023, Family classification based on tree representations for malware. In Proceedings of the 14th ACM SIGOPS Asia-Pacific Workshop on Systems, pp. 65-71.
15. Daeef, A.Y., Al-Naji, A. and Chahl, J., 2022. Features engineering for malware family classification based api call. Computers, 11(11), pp.1-15.
16. Bao, H., Li, W., Chen, H., Miao, H., Wang, Q., Tang, Z., Liu, F. and Wang, W., 2024. Stories behind decisions: Towards interpretable malware family classification with hierarchical attention. Computers & Security, 144, p.103943.
17. Abbasi, M.S., Al-Sahaf, H., Mansoori, M. and Welch, I., 2022. Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection. Applied Soft Computing, 121, p.108744.
18. Dabas, N. and Sharma, P., 2023. MalAnalyser: An effective and efficient Windows malware detection method based on API call sequences. Expert Systems with Applications, 230, p.120756.
19. Kattamuri, S.J., Penmatsa, R.K.V., Chakravarty, S. and Madabathula, V.S.P., 2023. Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence. Electronics, 12(2), pp.1-25.
20. Li, C., Lv, Q., Li, N., Wang, Y., Sun, D. and Qiao, Y., 2022. A novel deep framework for dynamic malware detection based on API sequence intrinsic features. Computers & Security, 116, pp.1-15.
21. Li, C. and Zheng, J., 2021. API call-based malware classification using recurrent neural networks. Journal of Cyber Security and Mobility, pp.617-640.
22. Catak, F.O., Yazı, A.F., Elezaj, O. and Ahmed, J., 2020. Deep learning based Sequential model for malware analysis using Windows exe API Calls. PeerJ computer science, 6, pp.1-23.
23. Han, W., Xue, J., Wang, Y., Huang, L., Kong, Z. and Mao, L., 2019. MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. computers & security, 83, pp.208-233.
24. Zhang, S., Wu, J., Zhang, M. and Yang, W., 2023. Dynamic malware analysis based on api sequence semantic fusion. Applied Sciences, 13(11), pp.1-16.
25. Fei, N., Gao, Y., Lu, Z. and Xiang, T., 2021. Z-score normalization, hubness, and few-shot learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 142-151.
26. Aldosari, F., Abualigah, L. and Almotairi, K.H., 2022. A normal distributed dwarf mongoose optimization algorithm for global optimization and data clustering applications. Symmetry, 14(5), pp.1-28.
27. Moustafa, G., El-Rifaie, A.M., Smaili, I.H., Ginidi, A., Shaheen, A.M., Youssef, A.F. and Tolba, M.A., 2023. An enhanced dwarf mongoose optimization algorithm for solving engineering problems. Mathematics, 11(15), pp.1-26.
28. Agushaka, J.O., Ezugwu, A.E., Olaide, O.N., Akinola, O., Zitar, R.A. and Abualigah, L., 2023. Improved dwarf mongoose optimization for constrained engineering design problems. Journal of bionic engineering, 20(3), pp.1263-1295.
29. Adwait Dathan, R. and Shanmuga Priya, S., 2022, Yield Forecast of Soyabean Crop Using Peephole LSTM Framework. In Proceedings of Third Doctoral Symposium on Computational Intelligence: DoSCI 2022 (pp. 261-270). Singapore: Springer Nature Singapore.
30. Tang, F., 2025. Short-term wind power prediction based on improved sparrow search algorithm optimized long short-term memory with peephole connections. Wind Engineering, 49(1), pp.71-90.
31. Ud. Chowdhury, Sohag Chakma. (2026). Impact of Pharmaceutical Residues on Endocrine Function and Reproductive Health in Estuarine Fish: Implications for Aquatic Animal Health and Sustainable Fisheries. *National Journal of Animal Health and Sustainable Livestock* , *4*(1), 1-8.
32. Shaik Sadulla, & K P Uvarajan. (2025). High-Level Synthesis-Driven Hardware/Software Co-Design for Reconfigurable Embedded AI Accelerators. *SCCTS Transactions on Reconfigurable Computing* , *3*(2), 49-55. https://doi.org/10.31838/RCC/03.02.06
33. A. Surendar. (2026). Designing Energy-Efficien t Embedded Systems Using Near- and Sub-Threshold VLSI Techniques. *National Journal of Advanced VLSI Design and Systems*, *1*(1), 9–16.
34. K. Geetha. (2025). Energy-Efficient Hardware Accelerator for Quantized Deep Neural Network Inference in Edge AI Applications. *Journal of Integrated VLSI and Signal Intelligence*, *1*(1), 18-25.