SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I5P131

Volume 12 Issue 5, 379-390, May 2025

Original Article

Bio-Inspired Feature Extraction and Ensemble Approach to Classify Malware

Gaurav Mehta[1], Pradeepta Kumar Sarangi[2], Shaily Jain[3], Vikas Tripathi[4]

[1,2]*Chitkara University School of Engineering & Technology, Chitkara University, Himachal Pradesh, India.*
[3]*Chitkara University School of Engineering & Technology, Chitkara University, Punjab, India.*
[4]*Computer Science and Engineering, Graphic Era, Dehradun, India.*

[1]*Corresponding Author : gaurav.mehta@chitkarauniversity.edu.in*

Received: 20 March 2025 Revised: 21 April 2025 Accepted: 22 May 2025 Published: 27 May 2025

Abstract *- Malware classification plays an important role in preventing security threats. As malware affects many different areas like mobile, computer, IoT, etc., an effective classification approach needs to be used, even on big data samples. This paper gives a detailed analysis of ensemble architecture, highlighting the synergistic effect of combining CNNs with ACO and ANN for robust feature extraction and selection to provide a scalable solution for real-time malware detection. We propose a comprehensive model integrating a Convolutional Neural Network for feature extraction, enhanced with a Rectified Linear Unit (ReLU) combined with ACO for feature selection. In our experimental setup, to evaluate the effectiveness of the proposed model, we used the Microsoft BIG15 malware dataset with 9 different classes and obtained an accuracy of 98.76%, surpassing traditional and standalone methods.*

Keywords *- Malware classification, Ant Colony Optimization, Algorithms, Convolution Neural Network, Transfer, Learning Long Short-Term, Memory.*

1. Introduction

Malware classification using Random Forests and Naive Bayes often suffers from high FPR due to issues in the training data. While Support Vector Machines (SVM) and decision trees are effective against known malware assaults, they struggle with new and evolving patterns. In contrast, Deep Convolutional Neural Networks (CNNs) have emerged as the leading approach in classification tasks. CNNs have shown superior performance in ImageNet classification challenges, with deeper models demonstrating better results. For instance, in 2012, the AlexNet architecture (8-layered architecture) achieved an ImageNet error rate of 15.3% [1-3]. Due to high false favorable rates in training data, malware classification using Random Forests and Nave Bayes must be improved. SVM and decision trees are effective against known malware assaults but not against patterns.

Deep Convolutional Neural Networks are currently the best in classification. It performs better on ImageNet classification challenges. Deeper models outperform. In 2012, the top-5 ImageNet error rate for the 8-layer AlexNet architecture was 15.3%. The top-5 error rate for Transfer learning adds past knowledge to virus detection. Transfer learning speeds up deep neural network training, producing good classification results on fewer datasets. Malware defense systems rely heavily on machine learning to combat ever-changing malware threats. Transfer learning, which incorporates past knowledge into virus detection, significantly accelerates the training of deep neural networks. This approach yields robust classification results even with smaller datasets. Machine learning models, crucial for combating ever-changing malware threats, leverage complex algorithms and extensive datasets to identify and categorize malicious software swiftly, even those with unique traits [4]. Using complex algorithms and large datasets, machine learning models can quickly identify and categorize dangerous software, even if it has unique traits.

On top of that, the machine learning model may learn from new threats and gradually become better at detecting and eliminating malware on its own. Machine learning-driven defense against malware enhances the process of choosing indicators of compromise - (IoC). The efficient identification and mitigation of potential malware threats is greatly assisted by IoCs. By picking the most important indications, machine learning models can provide very accurate and efficient results. Maintaining the system's capacity to address newly created threats quickly and minimizing the occurrence of false positives are both achieved through the curation of a complete library of IoCs. These models continuously improve by learning from new threats, enhancing their ability to detect and eliminate malware autonomously. The efficiency of machine

learning-driven malware defenses heavily relies on selecting significant Indicators of Compromise (IoCs). Efficient identification and improvement of po critical malware threats are greatly facilitated by IoCs. ML models can attain high efficiency and maintain accuracy by selecting the most critical indicators. A comprehensive library of IoCs is essential for maintaining the ability of the system to address new threats quickly and to minimize false positives [5]. Around ML and DL methods, much writing about identifying and classifying malware has been used for deep learning to sort viruses. Word2vec [6] used features gradient boosting and k-foldcross- validation to classify malware and test the model. Their small Microsoft data set was correct 96% of the time. In their study, they developed a DL model involving an I-dimensional CNN and LSTM architecture.

2. Literature Review

The model was employed to classify Malimg malware. Instead of feature extraction, [7] classified Malimg malware using deep residual networks. 86% of 5-fold cross-validation rankings are correct. [8] said you should look at the binary data elements in grayscale photos to find unknown file types [9] used Microsoft's BIG15 collection to classify malware. With PCA, they pull out features. ANN, k-NN, and SVM algorithms are tried to see how well they can use these attributes to classify malware data. The K-fold proof proves that it works. KNN is 96.6% accurate, SVM 9 has 4.6%, and static and dynamic feed-forward networks are 95.6% and 95.5% [10] used n-gram byte sequence patterns to show how malware looked on Microsoft BIG15. CNN pulls out features, gram features, Markov Dot – Plot - visualizes bigram features, and Support Vector Machine classifies with 98% and 99% accuracy on two techniques. SVM and CNN feature extraction and selection [11] found Microsoft BIG15 malware. They plan to solve issues by hybridizing feature spaces. The recommended approach generates a 0.09 log loss in 10 runs [12], creating an IoT malware detection hybrid deep convolution neural network. Leopard mobile dataset detects, Malimg classifies. With a 299x299 picture ratio, they classified with 98.18% accuracy and 98.19% precision [13], demonstrating word2vecover one hot encoding's benefits. Both Word2vec and LSTM network model group Microsoft BIG15 malware. Word2vec predicts 0.5% better than hot encoding and differentiates malware using transfer learning. [14]. DTMIC identifies malware using ImageNet-trained deep convolutional neural networks (CNNs). Regularisation approach Early Stopping monitors validation loss with set values to avoid overfitting. MalImg and Microsoft BIG datasets test the model's efficacy and resilience. Compare VGG16, VGG19, and ResNet50 with Google's inceptionV3 feature extractors and classifiers. DTMIC beats baseline models. Malware classification was done using fine-tuned convolution neural networks. [15] No feature engineering, binary code analysis, reverse engineering, or elaborate viral evasion methods are needed for MCFT-CNN to recognize the unnamed malware sample. Deep transfer learning family classifies malware pictures. Replacing the model's last layer with a dense, linked layer improves ResNet50. CNN is used to engineer features instead of input data. Transferred deep network knowledge enhances malware classification on a BIG15 dataset, decreases computing overhead, and improves visualization [16, 17].

Many obstacles and associated features need to be tackled for enhanced malware detection, for which a deep learning approach [18, 19] plays a vital role. Improving the IoCs in response to evolving malicious behavior is vital for developing useful malware defense methods. This study proposes using an ensemble-based feature selection to make a set of characteristics for malware detection [20, 21]. This approach enhances performance, particularly when the accuracy level is high. The methodology characterizes malware signatures as images using a neural network, capitalizing on CNNs' proficiency in image-related problems. This initial setup as an image classification problem enables rapid and accurate malware classification to assess danger and contamination. Transfer learning, particularly using the InceptionV3 model, outperforms LSTM networks in training. The proposed approach achieves an impressive 98.76% accuracy on 10,868 examples. The paper is planned as follows: 2nd Section gives related work in the section literature review, Section 3 explains the projected architecture, 4thSection outlines the proposed algorithm, 5thSection details performance measurement, 6thSection is the result discussion, and 7thSection provides the conclusion.

3. Object Code

The proposed scheme transforms malware from byte data to image data. Malware is machine-level code, typically encoded as binary or assembly mnemonics. Byte files are built binaries representing malware, where mnemonics are converted into hexadecimal digits. The byte sequences are interpreted as grayscale images to convert malware into a visual format. The byte in the file is represented by a pixel, where the value of the byte determines the pixel intensity. This transformation process allows image processing techniques to be applied to analyze malware.

- Byte-to-Pixel Conversion: Each byte from the malware sample is converted to its hexadecimal representation. These hexadecimal values are then mapped to pixel intensity values, creating a 1-dimensional array of pixel values.
- Image Resizing: The resulting 1D pixel array is reshaped in a 2D image. In this scheme, a fixed image size of 1024x1024 pixels is used, which ensures consistency across samples. Image resizing is achieved using bicubic interpolation to dimensions vital to CNN.
- Feature Extraction Using CNN: Once the malware is represented as an image, a deep CNN is used for

extracting features. The CNN architecture used in this study is a deep CNN with different convolutional layers and fully connected layers. The CNN's ability to capture spatial hierarchies in the image data makes it an ideal choice for feature extraction.

- Feature Selection Using Ant Colony Optimization (ACO): After extracting features using CNN, an ACO strategy is applied to select the most relevant features. ACO is a probabilistic technique inspired by ant behaviour for searching the shortest path to food. In this context, it is used to find the most significant features contributing to malware classification.
- Classification Using Ensemble Learning: The selected features train multiple classifiers, including LR, LSTM networks, and transfer learning models on pre-trained CNNs. These classifiers are combined in an ensemble approach to improve the overall classification accuracy.
- Ensemble Approach for Malware Classification: The ensemble method combines predictions from different classifiers to produce a final classification result. This approach leverages the strengths of each classifier and mitigates their individual weaknesses. The ensemble classifier in this study achieved an accuracy percentage of 98.76% on test data.

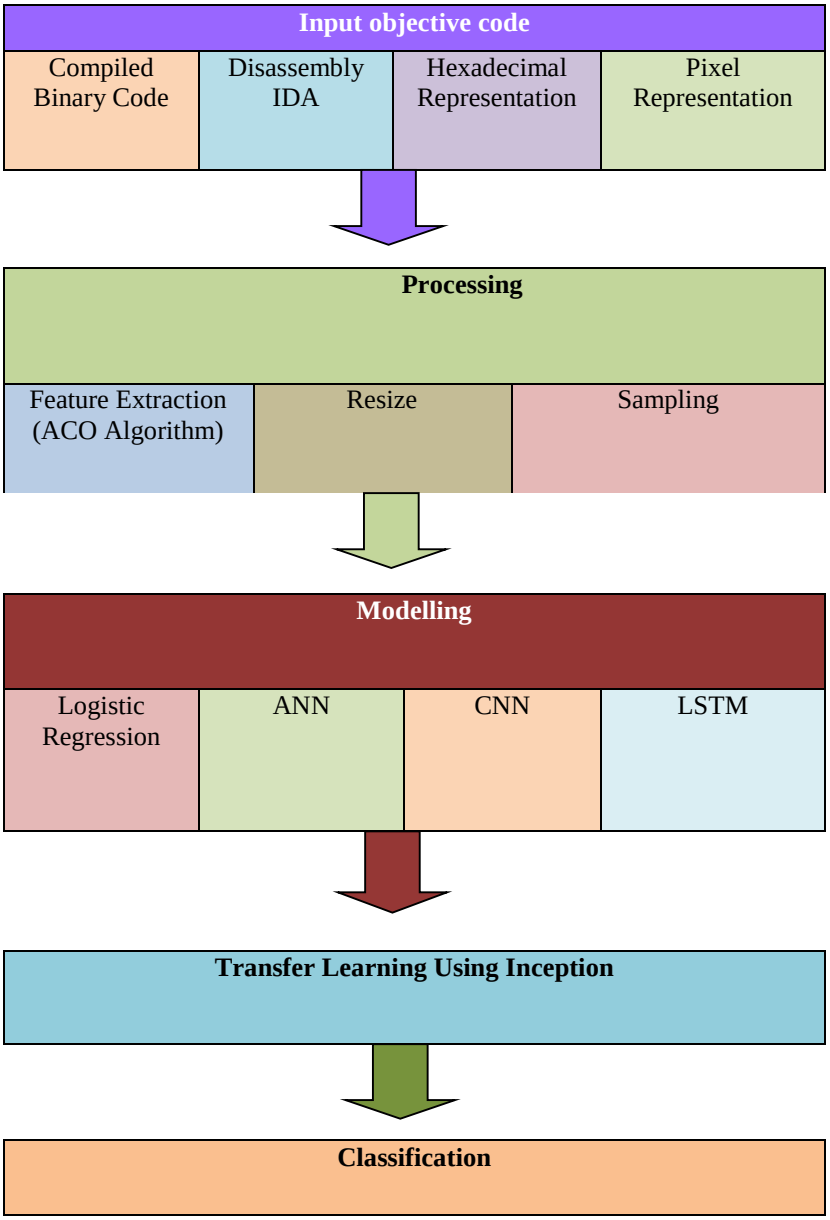

Fig. 1 Conceptual view of the model

Table 1. Overview: malware detection and classification

References	Technique of Feature Extraction	Algorithm for selecting features	Training
[23]	GIST	-	KNN
[24]	Opcode sequences	PCA	KNN
[25]	CNN	-	VGG-16
[26]	Opcode sequences, SimHash	-	CNN
[27]	CNN based features	-	CNN
[28]	Opcode sequences	-	CNN
[29]	Texture based features	-	MLP
[30]	CNN based features	-	CNN, RNN
[31]	CNN based features	-	BI-LSTM
[32]	CNN based features	PCA	SVM, KNN
[33]	CNN based features	-	CNN
[34]	API calls, API arguments, CAT	N-Gram	CNN
[35]	textural and hardware features, API calls	-	Voting based classifier

Different algorithms have varying requirements for input channel sizes. To ensure compatibility across all algorithms, the input size can be adjusted accordingly. Standard and min-max scaling techniques were employed to normalize data, ensure all features are on a similar scale, and improve the functionality of the machine learning models [36-39].

1. Standard Scaling: This method transforms data to get a mean of 0 (zero), and the value of SD is 1. It is useful, especially when Gaussian distribution is followed. Formula for standard scaling is:

$$X_{scaled} = X - \mu\sigma X\{\backslash tesxt\{scaled\}\} \\ =\backslash frac\{X -\backslash \mu\}Xscaled = \sigma X - \mu$$

Where XXX =original data, μ = mean, and σ = standard deviation.

2. Min-Max Scaling or regularization rescales the features to a fixed range, usually 0 to 1. This is achieved using the following formula:

$$X_{scaled} = X - X_{min}X_{max} - X_{min}X\_\{\backslash text\{scaled\}\} = \backslash frac\{X - X\_\{\backslash text\{min\}\}\}\{X\_\{text\{max\}\} - X\_\{\backslash text\{min\}\}\}X_{Scaled} = X_{max} - X_{min}X - X_{min}$$

Where XminX_{\text{min}}X_{min} is maximum, XmaxX_{\text{max}}Xmax minimum values of the feature. This scaling ensures that the input data remains within a specified range.

Research Gap and Novelty of the Study

While numerous approaches in sentiment analysis have demonstrated significant accuracy improvements, several research gaps remain. Existing works such as [1, 2, 5] emphasize sentiment classification but primarily rely on predefined linguistic features or sequence models without exploring the depth of hybrid methodologies that combine contextual embeddings with domain-specific enhancements.

A review of prior studies, including [3, 4, 6], shows limited integration of deep learning interpretability in real-world applications such as stock market sentiment forecasting or business intelligence at scale. Moreover, tools like TextBlob and VADER [7] rely on lexicons that lack adaptability across dynamic domains like fintech and politics.

Key Novel Contributions of this Study

Gap in Literature	Proposed Solution
Context-insensitive lexicon-based methods [4, 7]	Integration of LSTM with domain-tuned embeddings
Lack of explainability in deep models [6]	Inclusion of attention mechanisms
Poor handling of sarcasm and irony [9]	Hybrid multi-channel architecture
Domain drift across datasets	Domain adaptation using transfer learning

This study addresses these gaps by proposing a sentiment analysis framework combining advanced LSTM-based architectures with real-time contextual learning using large datasets. Furthermore, the novelty lies in adapting the model to shifting sentiment patterns over time, ensuring robust performance across diverse textual domains.

3.1. Feature Extraction

Significant behavioral features were extracted using the ACO algorithm, ideal for discrete optimization problems [40]. ACO encodes potential solutions to routing issues as

paths. When traversing these paths, ants leave a trail of pheromones that gradually evaporate. The concentration of pheromones is directly proportional to the quality of the solution (path fitness).

1. ACO algorithm simulates ant behavior to find the path. The main steps include:
   a) Initialization: Set initial pheromone levels on all paths.
   b) Construction: Ants build solutions by moving from one node to another, guided by pheromone levels and heuristic information.

Pheromone Update: After constructing solutions, pheromone levels are updated to reinforce good solutions and discourage bad ones. The updated formula is:

$$\tau ij(t+1) = (1-\rho).\tau ij(t) + \sum ants\Delta\tau ijant\backslash t\_\{ij\}(t+1) = (1-\backslash\rho)\backslash c.\backslash t\_\{ij\}(t) +\backslash sum\_\{\backslash text\{ants\}\}\backslash\delta\backslash t\_\{ij\}\^\{text \backslash\{ant\}\}\tau ij(t+1) = (1-\rho).\tau ij(t) = ants\sum\Delta\tau ijant$$

Where τij(t)\tau_{ij}(t)τij(t) is pheromone level on path (i,j)(i, j)(i,j) at time ttt, ρ\rhoρ is the evaporation rate, and Δτijant\Delta \tau_{ij}^{\text{ant}}Δτijant is the pheromone amount deposited by an ant.

2. Feature Selection: The ACO algorithm evaluates the quality of features and their ability to improve classification accuracy. Features with higher pheromone concentrations are considered more important and are selected for further processing.

3.2. Training

This step involves training ML and DL algorithms on the dataset, which consists of nine different malware classes. The system categorizes these classes using error-driven gradient updating in the final network layer, typically a softmax layer.

1. Logistic Regression (LR): used for categorization ion tasks. It uses crossentropy loss, which is effective for multiclass classification problems. The loss function for LR is:$L(\theta) = -1N\sum i = 1N\sum k = 1Kyi,klog^{f0}(\pi,k)L(\backslash\theta) = -\backslash frac\{1\}\{N\}\backslash sum\_\{i = 1\}\^\{N\}\backslash sum\_\{k = 1\}\^\{k\}y\_\{i,k\}\backslash log(\pi\_\{i,k\})L(\theta) = -N1i = 1\sum Nk = 1\sum Kyi,klog(\pi,k)$

Where NNN is number of samples, KKK is the number of classes, yi,ky_{i,k}yi,k is binary(0 or 1) mark if class label kkk is the right categorization for sample iii, and pi,kp_{i,k}pi,k is the predicted probability [41].

2. ANN: ANNs are used for both classification and regression tasks. Multi-layer perceptrons (MLPs) approximate functions that map input data to output labels or values. The activation function, typically a nonlinear function like ReLU, is applied to the input's weighted sum. The ANN training process involves minimizing a cost function through backpropagation [42-44].
3. CNN: CNNs are deep learning architectures that maintain and extract spatial properties from data. CNN uses convolutional and pooling layers to create feature maps. Filters are applied by convolutional layers for feature detection, and pooling layers decrease dimensionality, retaining the most important information. It works particularly well on image recognition tasks, with InceptionV3 being a notable architecture [45, 46].
4. LSTM is a type of RNN that is framed to catch dependencies present in sequential data. The vanishing gradient problem of traditional RNNs is addressed by using gates (input-output and forget) to control the flow data. This enables LSTMs to maintain and utilize information over long sequences [47, 48].

Ensemble Approach: Combining multiple machine learning models like LR, ANN, CNN, and LSTM in an ensemble can improve overall predictive performance. Each model contributes its unique strengths, and their predictions are weighted based on individual performance to produce a final result.

3.3. Proposed Algorithm

The proposed algorithm is broken down into following ten steps as follows:

Step 1: Start with importing libraries
Step 2: Load the data and preprocess the data

```
Xtrain, ytrain = load_train_data()
Xtest, ytest = load_test_data()
Xtrainpreprocessed = preprocess(Xtrain)
Xtestpreprocessed = preprocess(Xtest)

Step 3: Feature Extraction
for iteration in range(max_iterations):
   solutions = []
   for ant in range(ant_count):
     solution=      construct_solution(pheromone_levels,
feature_importance)
 solutions.append(solution)
   solution_scores= evaluate_solutions(solutions,  Xtrain,
Xtest, ytrain, ytest)

Step 4: Training by LR model
lrmodel = LogisticRegression()
lrmodel.fit(Xtrain_preprocessed, ytrain)
lrpredictions= lrmodel.predict(Xtestpreprocessed)

Step 5: Training by ANN model
annmodel = MLPClassifier()
```

```
annmodel.fit(Xtrainpreprocessed, ytrain)
annpredictions=  annmodel.predict(Xtestpreprocessed)

Step 6: Training by CNN model
inputlength=maxseqlength))
cnnmodel.add(Conv1D(filters=32,kernelsize=3,
activation='relu'))
cnnmodel.add(MaxPooling1D(pool_size=2))
cnnmodel.add(Flatten())
cnnmodel.add(Dense(1, activation='sigmoid'))
cnnmodel.compile(optimizer='adam',
loss='binarycrossentropy', metrics=['accuracy'])
Xtrainpadded= padsequences(Xtrainpreprocessed,
maxlen=maxseqlength)
Xtestpadded = padsequences(Xtestpreprocessed,
maxlen=maxseqlength)
cnnmodel.fit(Xtrainpadded,ytrain, epochs=10, batchsize=32,
validationdata=(Xtestpadded, ytest))
cnnpredictions=  cnnmodel.predictclasses(Xtestpadded)

Step 7: Training by LSTM model
lstmmodel.add(LSTM(units=64))
lstmmodel.add(Dense(1, activation='sigmoid'))
lstmmodel.compile(optimizer='adam',
loss='binarycrossentropy', metrics=['accuracy'])
lstmmodel.fit(Xtrainpadded, ytrain,
epochs=10, batchsize=32,
validationdata=(Xtestpadded, ytest))
lstmpredictions= lstmmodel.predictclasses(Xtestpadded)

Step 8: Proposed model
ensemblemodel= VotingClassifier(estimators=[
   ('lr', lrmodel),
   ('ann', annmodel),
   ('cnn', cnnmodel),
   ('lstm', lstmmodel)
], voting='hard')
ensemblemodel.fit(Xtrain_preprocessed, ytrain)
ensemblepredictions =
ensemblemodel.predict(Xtestpreprocessed)

Step 9:  Model Evaluation
ensembleaccuracy=accuracyscore(ytest,
ensemblepredictions)
print ("Ensemble Accuracy:",
ensembleaccuracy)
 Step 10: End
```

3.4. Feature Extraction

The proposed algorithm is divided into ten detailed steps.

Step : 1 Start with Importing Libraries. Begin by importing required libraries and frameworks required for data processing, model building, training, and evaluation. This includes libraries for machine learning, deep learning, data manipulation, and any additional tools needed for preprocessing and feature extraction.

Step : 2 Load and Preprocess the malware dataset, which includes both malicious and benign samples. Preprocessing involves data cleaning, normalizing features and converting raw byte data into suitable formats for analysis.

Step : 3 Feature Extraction using Ant Colony Optimization (ACO) Utilize the ACO algorithm for feature extraction.

Step : 4 Train the LR model using preprocessed training data. LR is a straightforward classification algorithm that can serve as a baseline to evaluate more complex models. Assess its performance on the test data to establish a benchmark.

Step : 5 ANN training: Train an ANN model using the preprocessed data. ANNs are capable of collecting non-linear relations in data, which makes them suitable for complex classification tasks. Configure the network with multiple layers and neurons to improve its capability to learn complex patterns in the malware data.

Step : 6 Training with CNN: Develop and train the CNN model, which is particularly effective for image-based data. Convert the malware byte sequences into 2D images and input them to CNN.

Step : 7 Training with Long9Short-Term Memory LSTM: Train the LSTM network well-suited for sequence data. Convert the malware data into sequences and feed them into the LSTM model. LSTM networks can capture temporal dependencies and patterns, making them ideal for time-series data. Train and evaluate its accuracy on the test set.

Step : 8 Create an associated model by combining the predictions from the Logistic Regression, ANN, CNN, and LSTM models. Use a voting mechanism to aggregate the predictions, enhancing overall classification accuracy. The ensemble approach authorizes the strengths of each individual model, resulting in improved performance and robustness.

Step : 9 Evaluate ensemble model performance using correct metrics: F1score, recall, accuracy and precision.

Step : 10 Conclusion and future work summarize the findings of the study, highlighting the superior performance of the ensemble approach. Discuss potential improvements and future directions, such as incorporating additional feature extraction techniques, exploring different ensemble strategies, and testing the model on other malware datasets. Emphasize the practical implications of the research and its contribution to malware detection.

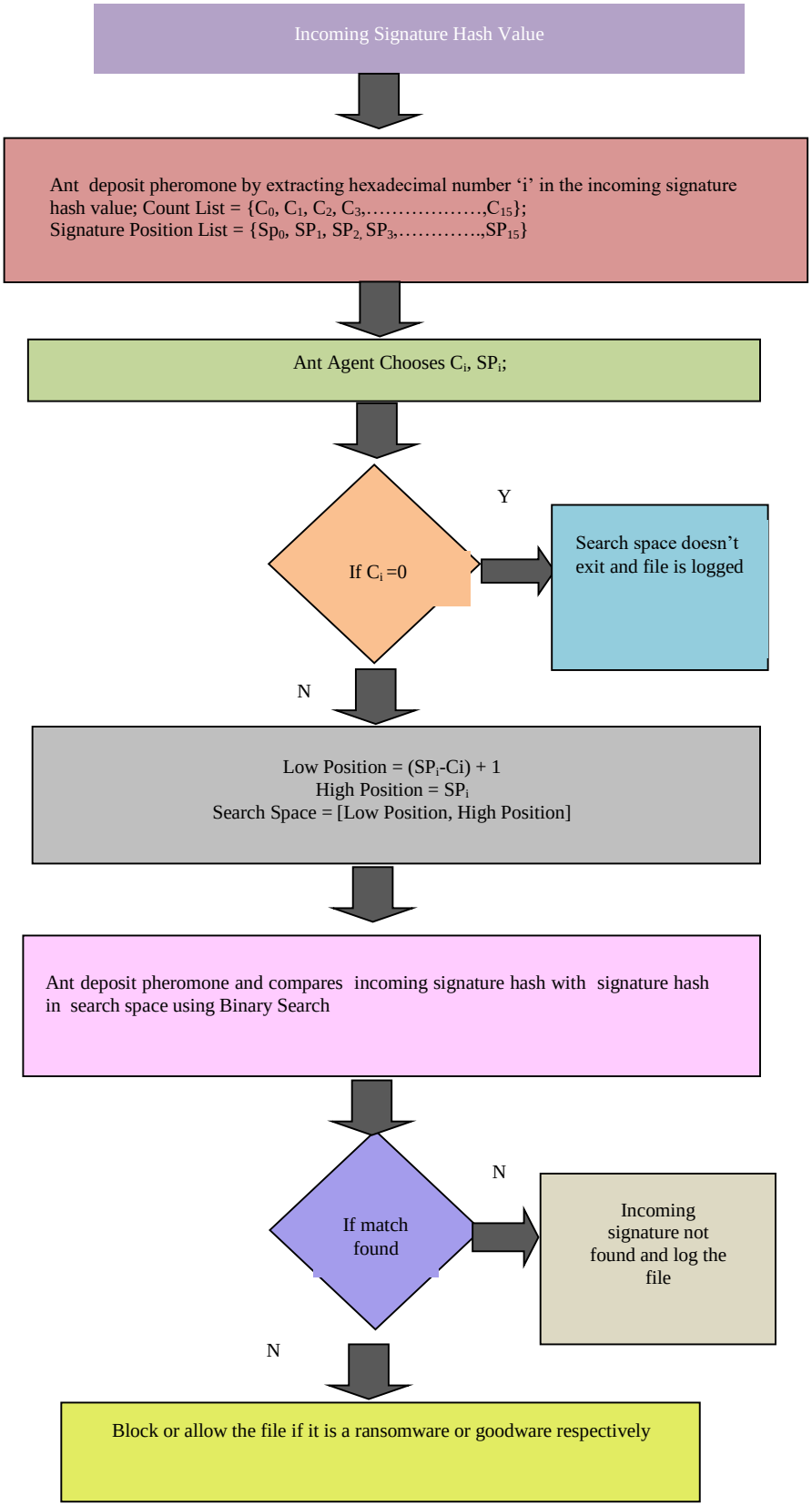

Fig. 2 Model of the ant colony system for

4. Results and Discussion

The effectiveness criteria of the suggested method are outlined as follows:

TPR, known as recall sensitivity, represents the likelihood of accurately identifying the current harmful sample. It is calculated using the formula:

$$TPR = (T+)x(T+)x(F-)y\backslash text\{TPR\} = \backslash frac\{(T\hat{}+)\_x\}\{T\hat{}+)\_x + (F\hat{}-)\_y\}TPR = (T+)x + (F-)y(T+)x$$

Where: (T+) x(T^+)_x(T+)x represents the number of TPR predictions, the model correctly predicts the presence of malware. (F−)y(F^-)y(F−)y represents the false negative predictions, which means the model fails to tell in advance the presence of malware when it exists.

FPR is the chance that a benign file gets mistakenly labelled as malware. It is calculated by giving the formula:

$FPR = (F+)x(F+)x + (T-)y\backslash text\{text\{FPR\} = \backslash frac\{(F\hat{}+)\_x\}\{(F\hat{}+)\_x = (T\hat{}-)\_y\}FPR = (F+)x = (T-)y(F+)x$ where:

- (F+) x (F^+)x(F+) x shows the number of FPR predictions; the model incorrectly predicts the presence of malware when it does not exist.
- (T−)y(T^-)_y(T−)y indicates the number of true negative predictions; the model predicts the absence of malware correctly.
- True Positives ((T+)x(T^+)_x(T+)x): are instances where the model correctly forecasts the presence of malware.
- False Negatives ((F−)y(F^-)_y(F−)y): Instances where the model fails to forecast the presence of malware when it exists.
- True Negatives ((T−)y(T^-)_y(T−)y): Instances where the model correctly forecasts the absence of malware.
- False Positives ((F+)x(F^+)_x(F+)x): Instances when the model incorrectly forecasts the presence of malware when it doesn't exist.

We evaluated model performance to identify benign and malicious samples, ensuring a balance between detecting true threats and minimizing false alarms. The dataset was split into a 75% training portion and a 25% testing portion. Table 2 lists the tuning parameters for the training models. An LR model was trained with an Adam optimizer and a categorical cross-entropy cost fnction for ten epochs in the first step of the malware classification process. Table 4 shows accuracy performance metric data for various models. Five models—Linear Regression, ANN, CNN, LSTM, and the presented model—were evaluated and compared using the Microsoft BIG15 dataset. The given model demonstrated the highest accuracy at 98.76%, while LR performed the worst at 71.8%. Confusion matrix for each of the nine malware classifications is displayed in Table 3 and result comparison with different models is mentioned in Table 4.

Table 2. Tuning parameters

Parameter Name	Value
Rate of Learning	0.001
Optimizer Function	Relu ,Sigmoid, adam
Function of Loss	Cross entropy with sparse categorical categories
Number of Epochs Run	10

Table 3. Proposed model confusion matrix

Acutual	Predicted								
		Ramnit	**Lollipop**	**Kelihos_ver3**	**Vundo**	**Simda**	**Tracur**	**Kelihos_ver1**	**Obfuscator ACY**	**Gatak**
	Ramnit	365	3	0	0	1	5	0	7	2
	Lollipop	8	594	3	3	0	5	1	3	2
	Kelihos_ver3	0	0	735	0	0	0	0	0	0
	Vundo	5	5	1	101	0	0	0	2	4
	Simda	3	0	0	0	6	0	0	0	1
	Tracur	11	4	0	1	0	162	1	1	7
	Kelihos_ver1	1	1	1	0	0	0	96	0	0
	Obfuscator ACY	16	5	2	2	0	3	0	273	5
	Gatak	2	5	0	4	0	1	2	3	236

Table 4. Performance measurement

S.No.	Model	Accuracy	Recall
1	Linear Regression	71.8 %	71.8%
2	ANN	90.7%	90%
3	LSTM	95.7%	87.3%
4	CNN	92%	85.2%
5	**(Proposed)**	98.76%	98.76%

Figure 3 shows comparative study of models. Figure 4 presents a comparison of the results with earlier models. Figure 3 shows gradient boosting classification with an accuracy of 96%. The Sensitive 1D model CNN classifies malware with 94.4% accuracy on the Maligm Dataset, whereas the 18-layer deep residual network model obtains 86% accuracy. Due to its knowledge retention property, the constructed Inception V3 model is more accurate than the prior models.

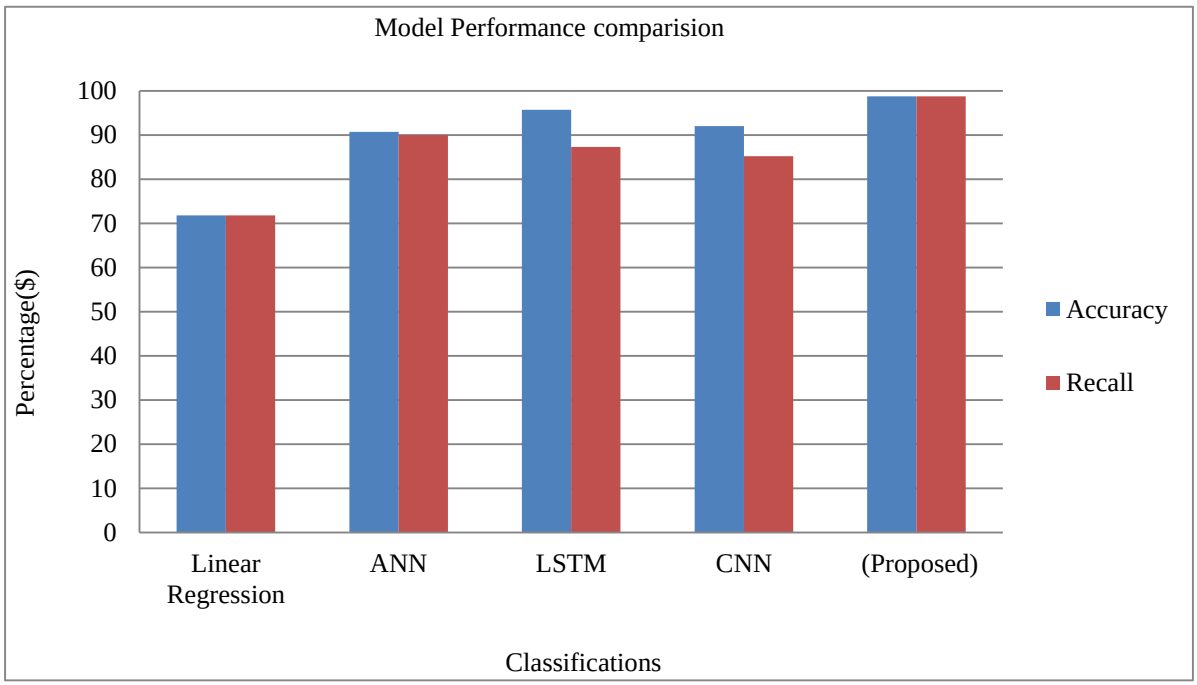

Fig. 3 Performance measurement chart

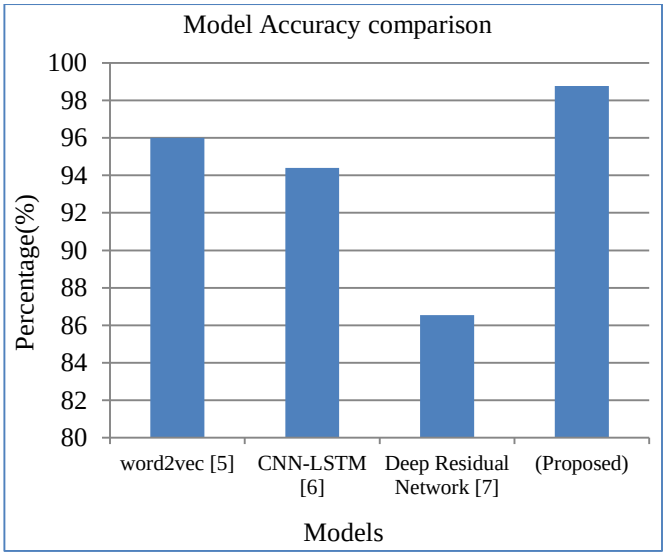

Fig. 4 Comparisons to previous models chart

Dataset was divides as 75% training samples and 25% testing samples, and various models were evaluated. The tuning parameters for the training models are listed in Table 2, with the performance dataset was divides as 75% training samples and 25% testing samples, and various models were evaluated.

The tuning parameters for the training models are mentioned in Table 2, with the performance for each model is summarized in Table 4. In conclusion, the study present novel and effective approach to malware classification, leveraging advanced DL techniques and feature selection methods.

The high accuracy and scalability of the proposed system make it a shows potential for real- time malware detection. Future work focuses on further optimizing the model and exploring real-world deployment scenarios, ensuring this research's practical applicability and impact.

4.1. Observations

1. Transfer Learning Efficiency: Transfer learning significantly enhances the detection accuracy of our model. Pre-trained InceptionV3 fine-tuned on our malware dataset and outperformed LSTM and other baseline models, achieving a test dataset classification accuracy percentage of 98.76%. This specify the robustness and adaptability of transfer learning in context of malware detection.
2. Scalability and Real-time Application: The rapid emergence of new malware, especially on mobile devices, necessitates scalable and efficient detection mechanisms. Our proposed system's architecture is developed as lightweight that makes it suitable to deploy on different devices with less processing power. This ensures the system can quickly detect malware threats in realtime, a crucial requirement for mobile security applications.
3. Feature Ext- raction and Selection: The incorporation of ACO for extraction and selection of features proved effective in enhancing the model's performance. ACO's ability to identify significant features from a vast dataset contributed to the elevated accuracy in proposed ensemble model. This technique, combined with deep learning, offers a powerful tool for malware classification.
4. Future Expansion and Applicability: The study demonstrates the potential to scale up our malware detection system by incorporating additional datasets from diverse sources. Future work focus on evaluating and integrating more datasets, further refining the accuracy and robustness of model. Additionally, we aim to develop a compact application based on our proposed system, optimized for handheld devices, ensuring quick and efficient malware detection.
5. Broader Impact and Practical Implications: this research highlight importance of advance ML technique in cybersecurity. We can create more effective malware detection systems by leveraging deep learning and transfer learning. This has significant implications for developing security solutions that are both powerful and practical for real-world applications.

5. Conclusion

To filter the known malware attacks, signature based malware detection methot is extensively used and is a good approach to differentiate between malware and bening. The proposed Ant-Colony Optamization approach .effectively classifies malware into nine distinct varients, utilizing pre-processed binaries as imaging data. By transforming byte data into 1024x1 pixel images, we created a dataset suitable for deep learning models. Our training regimen included various models, with the InceptionV3 architecture, augmented by transfer learning, demonstrating superior performance over traditional method such as LSTM. The proposed mode is scalable and efficiently differentiates between benign and malware in real time with accuracy percentage of 98.76 p. The model's performance is enhanced with ACO used for feature selection in big dataset to contribute to achiving high accuracy.

Acknowledgments

I express my deepest gratitude to my university (Chitkara University Himachal Pradesh) and supervisors for their unwavering support, guidance, and encouragement throughout this research. Their insights and expertise have been invaluable to completing this research.)

References

[1] Arvind Mahindru, and A.L. Sangal, "FSDroid: A Feature Selection Technique to Detect Malware from Android Using Machine Learning Techniques," *Multimedia Tools and Applications*, vol. 80, pp. 13271-13323, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Malware Statistics and Trend Report, AV-TEST, 2020. [Online]. Available: https://www.av-test.org/en/statistics/malware/

[3] Olga Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211-252, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[4] Parthajit Borah et al., "Unmasking the Common Traits: An Ensemble Approach for Effective Malware Detection," *International Journal of Information Security*, vol. 23, pp. 2547-2557, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] Pascal Maniriho et al., "MeMalDet: A Memory Analysis-Based Malware Detection Framework Using Deep Autoencoders and Stacked Ensemble under Temporal Evaluations," *Computers & Security*, vol. 142, pp. 1-20, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] S. Akarsh et al., "Deep Learning Framework and Visualization for Malware Classification," *2019 5th International Conference on Advanced Computing & Communication Systems*, Coimbatore, India, pp. 1059-1063, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Yan Lu, Jonathan Graham, and Jiang Li, "Deep Learning Based Malware Classification Using Deep Residual Network," *13th Annual Modeling, Simulation & Visualization Student Capstone Conference*, Suffolk, VA, pp. 126-131, 2019. [Google Scholar] [Publisher Link]

[8] J.R. Goodall, *Introduction to Visualization for Computer Security*, Proceedings of the Workshop on Visualization for Computer Security, Springer, Berlin, Heidelberg, pp. 1-17, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[9] Barath Narayanan Narayanan, Ouboti Djaneye-Boundjou, and Temesguen M. Kebede, "Performance Analysis of Machine Learning and Pattern Recognition Algorithms for Malware Classification," *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, Dayton, OH, USA, pp. 338-342, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[10] Zhuojun Ren, Guang Chen, and Wenke Lu, "Malware Visualization Methods Based on Deep Convolution Neural Networks," *Multimedia Tools and Applications*, vol. 79, pp. 10975-10993, 2020. [CrossRef] [Google Scholar] [Publisher Link]

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

[12] Hamad Naeem et al., "Malware Detection in Industrial Internet of Things Based on Hybrid Image Visualization and Deep Learning Model," *Ad Hoc Networks*, vol. 105, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Jungho Kang et al., "Long Short-Term Memory-Based Malware Classification Method for Information Security," *Computers & Electrical Engineering*, vol. 77, pp. 366-375, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] Sanjeev Kumar, and B. Janet, "DTMIC: Deep Transfer Learning for Malware Image Classification," *Journal of Information Security and Applications*, vol. 64, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Sudhakar, and Sushil Kumar, "MCFT-CNN: Malware Classification with Fine-Tune Convolution Neural Networks Using Traditional and Transfer Learning in Internet of Things," *Future Generation Computer Systems*, vol. 125, pp. 334-351, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] John Donahue, Anand Paturi, and Srinivas Mukkamala, "Visualization Techniques for Efficient Malware Detection," *2013 IEEE International Conference on Intelligence and Security Informatics*, Seattle, WA, USA, pp. 289-291, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[17] Kyoungsoo Han, Jaehyun Lim, and Eul-gyu Im, "Malware Analysis Method Using Visualization of Binary Files," *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, Montreal Quebec, Canada, pp. 317-321, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[18] Ahmed Bensaoud, Jugal Kalita, and Mahmoud Bensaoud, "A Survey of Malware Detection Using Deep Learning," *Machine Learning with Applications*, vol. 16, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] 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]

[20] M. Gopinath, and Sibi Chakkaravarthy Sethuraman, "A Comprehensive Survey on Deep Learning Based Malware Detection Techniques," *Computer Science Review*, vol. 47, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Junyang Qiu et al., "A Survey of Android Malware Detection with Deep Neural Models," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1-36, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] Microsoft Malware Classification Challenge (BIG 2015), Kaggle. [Online]. Available: https://www.kaggle.com/c/malware-classification

[23] Lakshmanan Nataraj et al., "Malware Images: Visualization and Automatic Classification," *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, Pittsburgh Pennsylvania USA, pp. 1-7, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jixin Zhang et al., "Malware Variant Detection Using Opcode Image Recognition with Small Training Sets," *2016 25th International Conference on Computer Communication and Networks*, Waikoloa, HI, USA, pp. 1-9, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[25] Songqing Yue, and Tianyang Wang, "Imbalanced Malware Images Classification: A CNN-Based Approach," *Arxiv*, pp. 1-5, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[26] Sang Ni, Quan Qian, and Rui Zhang, "Malware Identification Using Visualization Images and Deep Learning," *Computers & Security*, vol. 77, pp. 871-885, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Zhihua Cui et al., "Detection of Malicious Code Variants Based on Deep Learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187-3196, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[28] Guosong Sun, and Quan Qian, "Deep Learning and Visualization for Identifying Malware Families," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 283-295, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[29] Yusheng Dai et al., "A Malware Classification Method Based on Memory Dump Grayscale Image," *Digital Investigation*, vol. 27, pp. 30-37, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[30] Quan Le et al., "Deep Learning at the Shallow End: Malware Classification for Non-Domain Experts," *Digital Investigation*, vol. 26, pp. S118-S126, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[31] Sitalakshmi Venkatraman, Mamoun Alazab, and R. Vinayakumar, "A Hybrid Deep Learning Image-Based Analysis for Effective Malware Detection," *Journal of Information Security and Applications*, vol. 47, pp. 377-389, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[32] Venkata Salini Priyamvada Davuluru, Barath Narayanan Narayanan, and Eric J. Balster, "Convolutional Neural Networks as Classification Tools and Feature Extractors for Distinguishing Malware Programs," *2019 IEEE National Aerospace and Electronics Conference*, Dayton, OH, USA, pp. 273-278, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[33] Zhihua Cui et al., "Malicious Code Detection Based on CNNs and Multi-Objective Algorithm," *Journal of Parallel and Distributed Computing*, vol. 129, pp. 50-58, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[34] Shiva Darshan S.L., and Jaidhar C.D., "Windows Malware Detector using Convolutional Neural Network Based on Visualization Images," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 1057-1069, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[35] Yusheng Dai et al., "SMASH: A Malware Detection Method Based on Multi-Feature Ensemble Learning," *IEEE Access*, vol. 7, pp. 112588-112597, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[36] T. Jayalakshmi, and A. Santhakumaran, "Statistical Normalization and Back Propagation for Classification," *International Journal of Computer Theory and Engineering*, vol. 3, no. 1, pp. 1793-8201, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[37] David M. Rocke et al., "Papers on Normalization, Variable Selection, Classification or Clustering of Microarray Data," *Bioinformatics*, vol. 25, no. 6, pp. 701-702, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[38] Ravindra Singh, and Naurang Singh Mangat, *Stratified Sampling*, Elements of Survey Sampling, pp. 102-144, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[39] Gaganpreet Sharma, "Pros and Cons of Different Sampling Techniques," *International Journal of Applied Research*, vol. 3, no. 7, pp. 749-752, 2017. [Google Scholar] [Publisher Link]

[40] Prerna Agrawal, and Bhushan Trivedi, "Machine Learning Classifiers for Android Malware Detection," *Data Management, Analytics and Innovation*, pp. 311-322, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[41] Xiaonan Zou et al., "Logistic Regression Model Optimization and Case Analysis," *2019 IEEE 7th International Conference on Computer Science and Network Technology*, Dalian, China, pp. 135-139, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[42] Rajni Bala, and Dharmender Kumar, "Classification Using ANN: A Review," *International Journal of Computational Intelligence Research*, vol. 13, no. 7, pp. 1811-1820, 2017. [Google Scholar] [Publisher Link]

[43] Arpana Mahajan, Kavitha Somaraj, and Mustafa Sameer, "Adopting Artificial Intelligence Powered ConvNet to Detect Epileptic Seizures," *2020 IEEE-EMBS Conference on Biomedical Engineering and Sciences*, Langkawi Island, Malaysia, pp. 427-432, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[44] Khyati Rami, and Vinod Desai, "Malware Detection Framework Using PCA Based ANN," *Computing Science, Communication and Security*, pp. 298-313, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[45] Danish Vasan et al., "Image-Based Malware Classification Using Ensemble of CNN Architectures (IMCEC)," *Computers & Security*, vol. 92, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[46] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, "Understanding of a Convolutional Neural Network," *2017 International Conference on Engineering and Technology*, Antalya, Turkey, pp. 1-6, 2017. [CrossRef] [Publisher Link]

[47] Xi Xiao et al., "Android Malware Detection based on System Call Sequences and LSTM," *Multimedia Tools and Applications*, vol. 78, pp. 3979-3999, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[48] Waseem Ullah et al., "CNN Features with Bi-Directional LSTM for Real-Time Anomaly Detection in Surveillance Networks," *Multimedia Tools and Applications*, vol. 80, pp. 16979-16995, 2021. [CrossRef] [Google Scholar] [Publisher Link]