Cureus Journal of Computer Science

Open Access Original Article

Malware Classification Using Machine Learning and Deep Learning: A Comprehensive Approach

Ravin D [1] , Akshwin T [1] , Thenmozhi M [2]

1. School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, IND 2. School of Social Sciences and Languages, Vellore Institute of Technology, Vellore, IND

Corresponding author: Thenmozhi M, thenmozhi.nisha@vit.ac.in

Received 04/19/2025
Review began 04/24/2025
Review ended 06/30/2025
Published 07/04/2025

DOI: https://doi.org/10.7759/s44389-025-05024-y

Abstract

This research work investigates comprehensive machine learning and deep learning techniques for malware classification, addressing the limitations of traditional signature-based detection methods. By leveraging both static and dynamic features, we compare the performance of various classifiers like Decision Trees, Random Forest, XGBoost, Artificial Neural Network, and an Attention-Based Artificial Neural Network model. Experimental results reveal that Attention-Based Artificial Neural Network provides the highest accuracy, achieving 0.97, outperforming both traditional classifiers and standard ANN models in malware detection. Random Forest and XGBoost classifiers also showed strong results among traditional classifiers with accuracies of 0.95 and 0.96, respectively. These findings underscore the effectiveness of using deep learning, particularly attention mechanisms, in identifying complex patterns within malware behaviours, offering a significant improvement over traditional machine learning approach. The study aims to show that the Attention-Based Artificial Neural Network is highly effective for malware detection, presenting a scalable, accurate solution for identifying new and obfuscated malware types, which hold promise for future applications in cybersecurity.

Categories: AI applications, Deep Learning, Machine Learning (ML)
Keywords: malware classification, machine learning, deep learning, ensemble techniques, attention mechanism

Introduction

In the modern era, the rise of malware has become a significant threat to cybersecurity, affecting individuals, organizations, and governments worldwide. Malware, a term for malicious software, is designed to infiltrate and damage computer systems without the user's consent. With the increasing dependence on digital infrastructure, attacks by malware have evolved in complexity, from basic viruses to sophisticated forms such as ransomware, spyware, and advanced persistent threats. These malicious programs can cause financial loss, data breaches, and system disruptions, making their detection and classification a critical aspect of modern cybersecurity. For identifying malware, traditional malware detection methods, such as signature-based detection and heuristic analysis, have been used widely [1]. However, these approaches face several challenges, especially as malware becomes more complex and capable of evading detection through techniques like polymorphism, obfuscation, and zero-day exploits.

Signature-based methods rely heavily on known patterns, rendering them ineffective against new and unseen malware variants. Similarly, heuristic approaches, while more flexible, are prone to false positives and often struggle to adapt to rapidly changing attack vectors. These limitations have driven the search for more robust, scalable, and adaptive solutions. Machine learning (ML) and deep learning (DL) techniques offer better alternatives for malware detection and classification by learning patterns and behaviours from large datasets, enabling the identification of both known and unknown malware types [2]. ML models can automatically extract features from raw data, improving scalability and reducing the need for intervention of humans.

DL models have demonstrated impressive capabilities in handling complex data structures, such as byte sequences and application programming interface (API) calls, providing higher accuracy in malware classification tasks [3]. The primary objective of this paper is to explore and compare the effectiveness of ML and DL techniques in classifying malware. By leveraging various ML and DL models, we aim to develop an efficient framework capable of accurately detecting and categorizing different types of malwares, including both known and emerging threats. This study seeks to address the limitations of traditional methods by demonstrating the superior adaptability and precision of ML- and DL-based approaches in combating modern cyber threats.

Materials And Methods

Literature work

Bugra and Erdogan [3] presented a novel malware representation method utilizing static analysis,

specifically focusing on sequences of opcodes without arguments, and demonstrated the efficacy of this approach by employing a shallow DL-based feature extraction technique known as Word2Vec. Their work highlights the capabilities of the gradient boosting algorithm for the classification task, achieving an impressive accuracy of up to 0.96 through the implementation of k-fold cross-validation to validate model performance effectively. The results indicate that even with limited sample data, the proposed model can reach high classification accuracy. The authors also suggest that increasing the k-value in cross-validation may lead to even better results. They further emphasize the importance of hyperparameter optimization in gradient boosting machine, advocating tree pruning via grid search to enhance model accuracy. Looking ahead, they aim to extend their dataset to encompass all malware classes and explore the semantic relationships between malware opcodes as future research directions.

Pascanu et al. [4] explored the use of recurrent neural networks (RNNs), particularly networks, for malware classification based on sequences of API calls made by executable files. This approach harnesses the temporal pattern recognition capabilities of RNNs, enabling detection of malware based on dynamic behaviours rather than static features like code signatures. The evaluation of the model was done using metrics such as precision, recall, F1-score, and accuracy, with the model achieving an accuracy of 0.98 in identifying malware, showing its robustness against obfuscation techniques and its ability to generalize to unseen malware samples. The RNN-based model showed superior performance over traditional ML approaches, though challenges such as extended training times, interpretability issues, and performance drops with very long input sequences were noted. Despite these drawbacks, the proposed method excelled in detecting new malware variants and demonstrated strong evaluation metrics across different malware families.

Kalash et al. [5] investigated the application of deep convolutional neural networks (CNNs) for malware classification by transforming malware binaries into grayscale images and using CNNs to analyse these visual patterns. This innovative approach leverages the image recognition capabilities of CNNs to classify malware based on the structural characteristics of the binary code, offering an alternative to traditional feature-engineering methods. The authors evaluated their model using metrics such as accuracy, precision, recall, and F1-score, achieving an impressive accuracy of 0.9852 on the Malimg dataset, which contains over 9,000 malware samples from 25 different families. The CNN-based model demonstrated robustness in handling large-scale datasets and efficiently recognizing complex patterns across different malware families. However, limitations included the need for substantial computational resources and the potential for the model to misclassify sophisticated obfuscated malware. Despite these challenges, the CNN approach showed promising results, outperforming traditional methods in terms of accuracy and generalization across malware types.

Nari and Ghorbani [6] presented a methodology for classifying malware by analyzing its network behaviour. Instead of relying on static features such as binary signatures, the authors propose examining dynamic features derived from network traffic generated by malware. They capture network behaviour metrics like packet size, connection patterns, and communication frequencies to build a robust feature set for classification. Using ML techniques such as decision trees and support vector machines (SVMs), the study achieves promising results, demonstrating the effectiveness of network-based features in detecting malware, particularly when dealing with polymorphic or obfuscated samples. The model's accuracy was reported to be 0.971 on a dataset of captured network traffic, with metrics such as precision, recall, and F1-score used to assess performance. One of the key advantages of this approach is its resilience to evasion techniques that focus on altering file-level attributes. However, the system's dependence on capturing network activity means it can be less effective in offline environments or when malware operates in stealth mode, limiting network communication. Despite these limitations, the method shows potential in improving the detection of sophisticated malware in real-world environments.

Aslan and Yilmaz [7] introduced a novel framework for the malware classification using DL techniques. The authors leverage CNNs to automatically learn features from malware binaries converted into grey-scale images, bypassing the need for manual feature extraction. This image-based approach allows the model to detect patterns in malware behaviour that are difficult to identify using traditional techniques. The framework was evaluated using a comprehensive dataset, where the DL model achieved an accuracy of 0.99. The model demonstrated the high performance of the method in distinguishing between benign and malicious software samples. One major advantage of this approach is its ability to generalize well across different types of malwares, including polymorphic and metamorphic variants, thanks to the model's ability to extract intricate, non-linear features. However, a potential limitation is the computational complexity associated with training DL models, as well as the reliance on large datasets to achieve high accuracy. Despite these challenges, the framework presents a significant advancement in automated malware classification, offering both scalability and high detection rates.

Xue et al. [8] presented a novel approach to malware classification by combining probability scoring with ML techniques. The authors propose a probability scoring model to evaluate the likelihood of a software sample being malware, using features extracted from its behaviour and code structure. This scoring system includes traditional ML classifiers, such as Support Vector Machines classifier and Random Forest Classifier, to enhance the classification performance. The framework is evaluated using a comprehensive

malware dataset, attaining a high accuracy of 0.97, with precision, recall, and F1-score metrics indicating the system's robust ability to differentiate between benign and malicious software. The advantage of this approach is its ability to provide a probabilistic interpretation of the classification results, which adds an additional layer of transparency to the decision-making process. Moreover, combining probability scoring with ML significantly enhances the system's generalizability across various malware types. However, the model's performance could be influenced by the choice of feature extraction methods and the need for large, diverse datasets to improve accuracy. Despite these challenges, the proposed approach provides a promising direction for improving malware classification using probabilistic and ML techniques.

According to Al-Janabi et al. [9], malware, or malicious software, represents one of the most dangerous cyber threats, with attackers developing various forms of malware such as viruses, spyware, bots, and ransomware. The primary goal of malware is to harm systems or gain unauthorized access, and while many methods have been proposed to counter these threats, signature-based detection methods remain limited, particularly in addressing zero-day attacks and polymorphic viruses. ML-based malware detection has gained recognition as a modern and effective alternative. ML detection techniques are typically categorized into static, dynamic, and hybrid analyses. This study surveys different feature extraction and classification methods, reviewing representative research in the field and comparing their accuracy in detecting malware. It identifies the J48 algorithm and hybrid analysis as top performing techniques, achieving 1.0 accuracy in malware detection within Windows systems. Similarly, in Android systems, 1.0 accuracy was obtained using a decision tree algorithm with dynamic analysis. The work emphasizes the potential of these ML approaches and serves as a foundation for future research on malware analysis using ML methods.

Yadav et al. [10] have contributed significantly to the field of Android malware detection using computer vision methods, leveraging the advancements in CNNs. They proposed an image-based method where local and global features are extracted from grayscale malware images to classify Android applications into benign or malicious categories, achieving an accuracy of 0.968. Their work aligns with the growing interest in utilizing image-based representations for malware detection due to their effectiveness in handling code obfuscation and eliminating the need for manual feature engineering. Similar studies have highlighted the use of grayscale and RGB image representations for Android malware detection. These methods emphasize that image-based approaches can effectively capture complex malware behaviour patterns without relying on platform-specific characteristics. Additionally, CNN architectures like VGG16, ResNet50, and Efficient Net have shown remarkable performance, with Efficient Net achieving up to 0.95 accuracy. The literature underscores that CNN-based models outperform other ML classifiers like SVM and random forest when combined with image-based representations, highlighting their potential for more efficient, automated malware detection in real-world applications.

Ravi et al. [11] highlighted the increasing prevalence of malware attacks due to the rapid adoption of smart healthcare systems, emphasizing the critical need for secure and reliable malware detection mechanisms in these ecosystems. Their literature survey indicates that existing malware detection techniques primarily utilize DL approaches, leveraging features extracted from portable executable (PE) files such as PE-Headers, PE-Imports, PE-Images, and API calls. Recognizing the importance of these diverse feature sets in enhancing malware detection rates, they proposed a multi-View attention-based DL framework. Their experimental analysis demonstrated that the proposed method significantly outperformed both traditional ML and non-attention-based approaches, achieving an overall accuracy of 0.95 in malware detection. Moreover, the framework yielded even higher accuracy rates, with 0.98 for Windows-based datasets and 0.97 for Android-based datasets, illustrating its robustness and generalizability across different platforms. This research underlines the effectiveness of combining various feature sets for malware detection in smart healthcare systems, paving the way for future advancements in cybersecurity.

Ganesan et al. [12] address the urgent need for robust malware detection systems considering the growing number of malware attacks and the ongoing risk of zero-day exploits. Their research highlights the shift from traditional malware analysis methods that rely on byte information to image-based intrusion detection systems, which leverage the strengths of CNNs and attention mechanisms. By employing a residual attention network, their approach effectively captures significant structural features of malware, allowing for more precise identification and differentiation from benign files. This focus on relevant sections of malware not only enhances detection accuracy but also significantly reduces false positives, a crucial factor in cybersecurity accessibility. Their proposed method demonstrates impressive results, achieving an accuracy of 0.992, thereby outperforming conventional ML algorithms and existing CNN-based methods. This work underscores the potential of advanced DL techniques, such as attention mechanisms, in enhancing malware detection frameworks. Table *1* represents the literature survey on various existing methodologies.

Existing methods in the literature exhibit several limitations that are effectively addressed by the proposed attention-based approach. Traditional ML models such as SVM, gradient boosting machine, and random forest [3,6,8] heavily rely on manual feature engineering and lack the capability to capture spatial or sequential dependencies inherent in complex data. Recurrent models like RNN and long short-term memory [4] struggle with long-range dependencies, suffer from vanishing gradients, and are

computationally expensive due to their sequential nature. While CNN-based models [5,7,10,12] achieve high accuracy, they primarily focus on local features and treat all spatial regions equally, often missing global contextual information. Multi-model and hybrid systems [8,9] tend to be computationally intensive, prone to overfitting, and lack interpretability. Multi-view and residual attention networks [11,12] improve performance but often introduce high model complexity and training challenges. In contrast, the proposed attention mechanism selectively focuses on the most informative parts of the input, effectively capturing both local and global features, improving model interpretability, and reducing computational overhead while maintaining or enhancing classification accuracy.

Ref. No.	Methodology	Accuracy
[3]	Word2Vec+Gradient Boosting Machine	0.96
[4]	RNN +LSTM	0.98
[5]	Deep CNN	0.98
[6]	SVM	0.97
[7]	CNN	0.99
[8]	SVM+Random Forest	0.97
[9]	Static, Dynamic, and Hybrid Machine Learning Models	1.00
[10]	CNNs	0.97
[11]	Multi-View Attention-Based Deep Learning	0.95
[12]	CNNs and Residual Attention Network	0.99

TABLE 1: Literature Survey on Various Methodologies Used for Malware Detection and Their Accuracies

CNN, Convolutional Neural Network; LSTM, Long Short-Term Memory; RNN, Recurrent Neural Network; SVM, Support Vector Machine

Experimental analysis

System Configuration

All experiments were conducted on a system equipped with an NVIDIA Tesla T4 GPU, 32 GB RAM, and an Intel Xeon processor, using Python 3.10 with libraries including TensorFlow 2.11, Scikit-learn 1.3, and NumPy 1.24. The models were trained and evaluated using Kaggle to ensure consistent compute availability. This configuration enabled efficient training of DL architectures within reasonable time constraints, while also ensuring reproducibility across experiments.

Dataset

The Microsoft Malware Classification Challenge (BIG 2015) was a competition organized by Microsoft to promote the development of ML models for malware detection. The challenge is aimed at classifying malware samples into different families based on their file content and behaviour. Participants were provided with a dataset consisting of two parts. They are hexadecimal representations of the malware files and disassembled code information. The task involved creating models that could accurately assign one of nine malware families to each sample, leveraging both ML and DL analysis techniques. The dataset contained more than 20,000 labelled samples, categorized into families such as Ramnit, Lollipop, Kelihosver3, and others. Key issues addressed by participants included handling large, complex datasets, extracting meaningful features from raw binary data, and building robust classification models that could generalize to unseen malware types. The dataset is from the open data source [13]. Table *2* represents different classes in the dataset.

Class Number	Name of the Malware	Data Count
Class 1	Ramnit	1,541
Class 2	Lollipop	2,478
Class 3	Keilhos_ver3	2,942
Class 4	Vundo	475
Class 5	Simda	42
Class 6	Tracur	751
Class 7	Keilhos_ver1	398
Class 8	Obfuscator.ACY	1,228
Class 9	Gatak	1,013

TABLE 2: Various Classes in the Dataset

Statistical analysis shows that the dataset comprises a total of 10,868 malware samples, each represented by 68 extracted features along with a target label indicating the malware family, resulting in a dataset shape of (10,868, 69). A preliminary inspection confirms that there are no missing or null values, ensuring data quality and consistency for model training. The class distribution reveals a noticeable imbalance across the nine malware classes, with Class 3 (2,942 samples), Class 2 (2,478 samples), and Class 1 (1,541 samples) being the most prevalent. In contrast, Class 5 is significantly underrepresented, with only 42 samples.

Pre-Processing

We transformed the raw .asm and .bytes files into a one-hot encoded format, allowing ML models to work with the data directly without requiring extensive manual feature engineering. In standard malware analysis, these files typically demand intricate parsing to extract meaningful patterns, such as opcode sequences, instruction behaviours, or byte-level signatures indicative of malicious activity. This preprocessing approach simplifies this by converting categorical attributes into structured, machine-readable vectors. Specifically, elements like opcodes (e.g., mov, add, jmp), section identifiers (such as .text and .data), and hex byte values from the .bytes files were identified and encoded. Each distinct categorical item was mapped to its own binary column, resulting in a high-dimensional but sparse feature space. Files with corrupted or incomplete data were either excluded or filled with placeholder values to maintain consistency. By preparing the data in this way, we enable more efficient training of ML models and streamline the detection of malware traits within structured input. We have used 80% of the data for training and 20% for testing.

Dimensionality Reduction Using Autoencoders

The autoencoder is used as a tool for dimensionality reduction to manage the large number of features in the one-hot encoded malware dataset. The model’s encoder compresses high-dimensional one-hot data into a compact, lower-dimensional form (32 dimensions here). This step preserves essential information by focusing on meaningful patterns in the data, allowing the model to ignore irrelevant or redundant details. The autoencoder then tries to reconstruct the original high-dimensional input from this compact representation, learning to encode crucial features while discarding noise [14].

By minimizing reconstruction errors, it effectively captures the structure of the malware data. After training, only the encoder part of the autoencoder is retained and applied to both training and test sets, transforming them into their compressed, 32 dimensional representations. These encoded versions retain core characteristics of each malware sample but at a fraction of the original feature count, enabling efficient and faster downstream analysis. Figure *1* shows the architecture of autoencoders used in this case. The same underlying models - logistic regression, decision tree, and random forest - were employed for the ensemble model, with logistic regression acting as the meta-model in a stacking ensemble setup. Joblib was used to save the trained stacking model. Following training, the model's performance was assessed using a confusion matrix to visualize the classification results, accuracy, precision, recall, and F1-score. Figure *1* represents the architecture diagram of autoencoders used for dimensionality reduction.

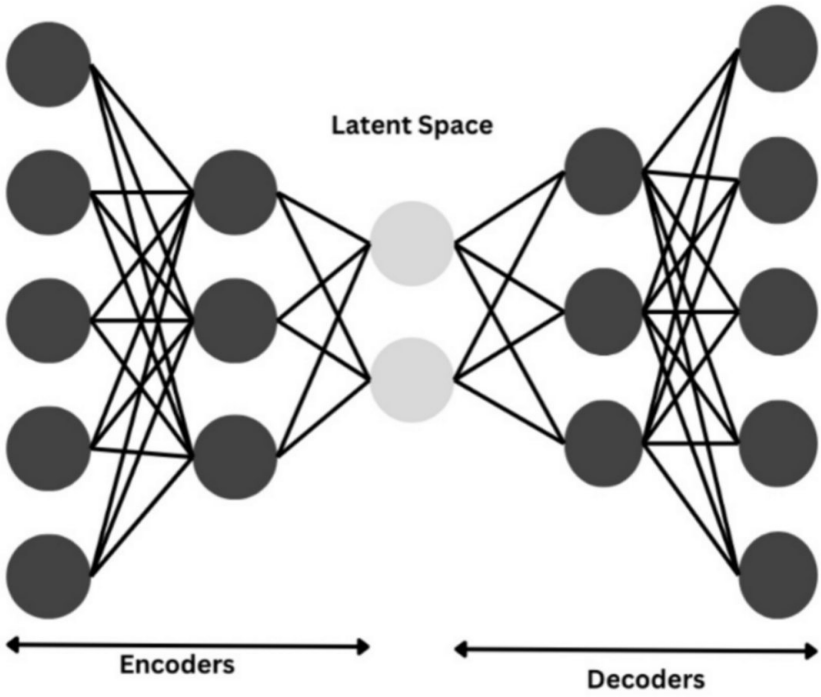

FIGURE 1: Architecture of Autoencoders for Dimensionality Reduction

ML models

Decision Tree Classifier

The decision tree classifier is a type of supervised ML model that can be applied to both classification and regression problems. It functions by repeatedly dividing the feature space into smaller subsets according to decision rules derived from the training data. When used for malware classification, the decision tree algorithm organizes the data based on feature values, creating a flowchart-like structure. In this structure, each internal node signifies a decision, and each leaf node represents a class label [15]. In this research, the decision tree classifier is utilized with the Gini impurity criterion to assess the quality of splits at each node. The model was initialized with a random state of 1 to ensure reproducibility and was trained on the encoded feature set of the malware dataset. The encoded data was divided into training and test sets to evaluate the model's ability to generalize. After training, predictions were made on the test set, and the accuracy of the model was calculated. The model achieved an overall accuracy of 0.947 on the test data. The classifier's performance was further evaluated using various metrics such as precision, recall, and F1-score for each malware class. Table *3* represents the evaluation metrics of different classes for decision tree classifier algorithm.

Class	Precision	Recall	F1-Score
Class 1	0.89	0.89	0.89
Class 2	0.98	0.98	0.98
Class 3	1.00	1.00	1.00
Class 4	0.92	0.90	0.91
Class 5	0.22	0.33	0.27
Class 6	0.96	0.90	0.93
Class 7	0.94	0.99	0.96
Class 8	0.88	0.89	0.88
Class 9	0.94	0.94	0.94

TABLE 3: Evaluation Metrics of Different Classes for Decision Tree Classifier Algorithm

From Table 3, it is important to note that Class 5 had the lowest performance with a precision of 0.22 and an F1-score of 0.27, likely due to a smaller number of samples.

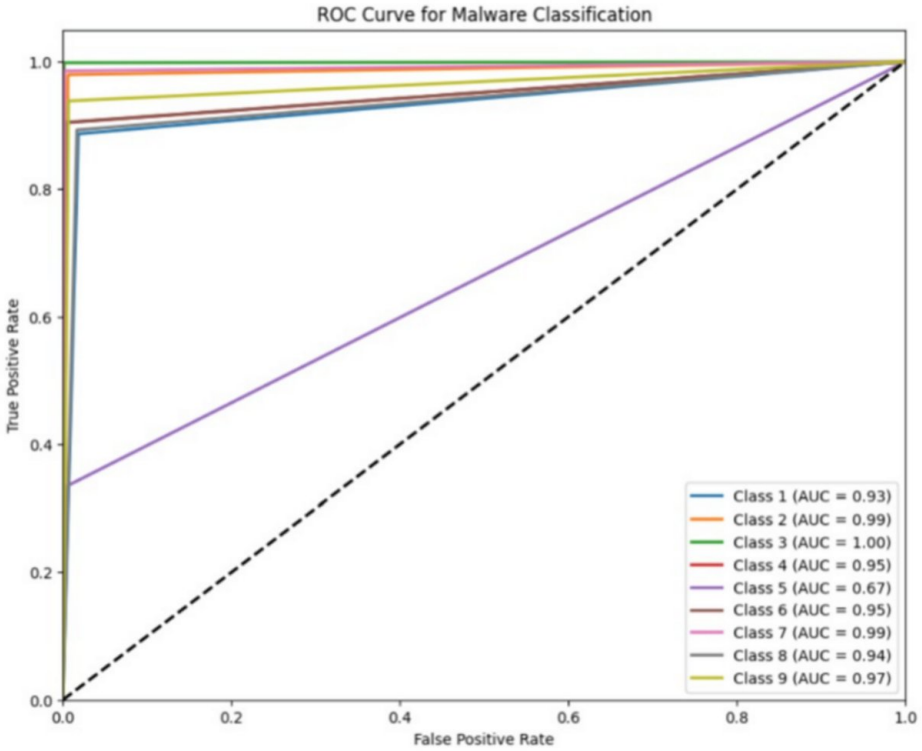

FIGURE 2: Receiver Operating Characteristic (ROC) Curve of the Decision Tree Classifier for Each Class

AUC, Area Under the Curve

Figure 2 represents the receiver operating characteristic (ROC) curve of the decision tree classifier for each class. To further assess the model, ROC curves were plotted for each class to evaluate the trade-off between the true positive rate (recall) and the false positive rate. The area under the curve (AUC) for each class ranged from 0.89 to 1.00, demonstrating the decision tree's strong discriminative performance across all malware types. The ROC curves for the 10 malware classes indicated high sensitivity and specificity, with AUC values close to 1.0 for most classes. Notably, Class 3 achieved a perfect AUC of 1.00.

Random Forest Classifier

Random forest classifier is an ensemble learning method that creates multiple decision trees and combines their results to enhance prediction accuracy and mitigate over-fitting [16]. The random forest model was configured with 50 decision trees, and each tree was allowed to consider up to 12 features during splitting. The random seed was fixed for reproducibility. The model was trained on the malware classification dataset using the encoded training features. After training, the random forest model was tested on the encoded test set, achieving an overall accuracy of 0.959. Table *4* represents the evaluation metrics of different classes for random forest classifier algorithm.

Class	Precision	Recall	F1-Score
Class 1	0.95	0.92	0.93
Class 2	1.00	0.98	0.99
Class 3	1.00	1.00	1.00
Class 4	0.98	0.95	0.96
Class 5	1.00	0.33	0.50
Class 6	0.99	0.90	0.94
Class 7	0.99	1.00	0.99
Class 8	0.99	0.91	0.95
Class 9	0.99	0.97	0.98

TABLE 4: Evaluation Metrics of Different Classes for Random Forest Classifier Algorithm

From Table *4*, it is seen that Class 5 has the lowest recall and F1-score, which are 0.33 and 0.50, respectively, which may be attributed to the small number of samples in this class. This class imbalance likely affected the model's ability to generalize for Class 5.

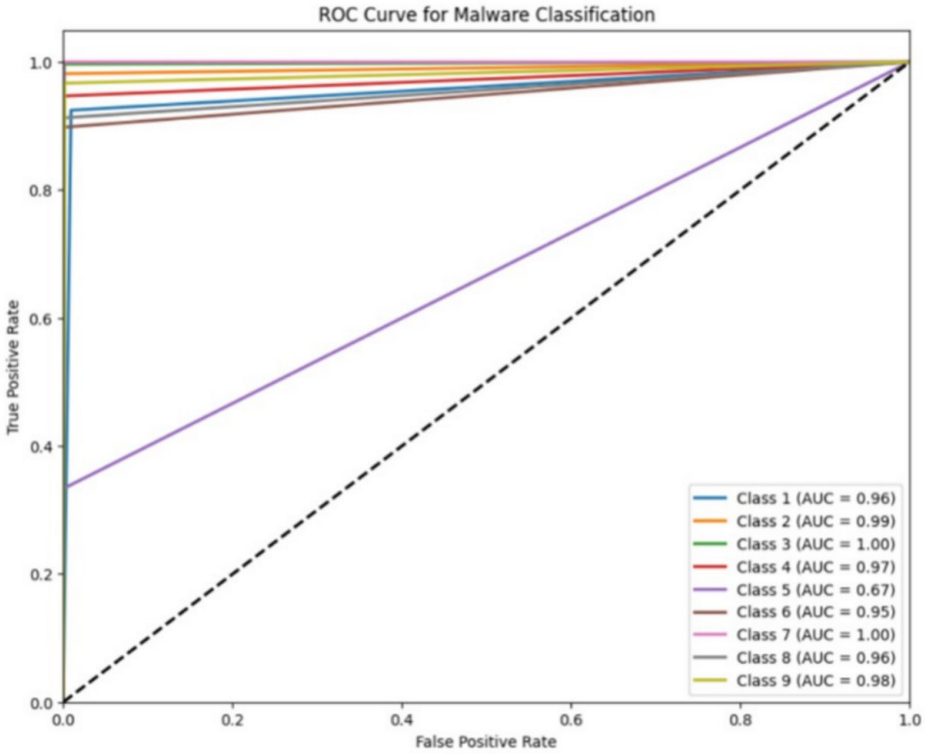

FIGURE 3: Receiver Operating Characteristic (ROC) Curve of the Random Forest Classifier for Each Class

AUC, Area Under the Curve

Figure *3* represents the ROC curve of the random forest classifier for each class. ROC curves were generated for each class to assess the trade-off between true positive rate and false positive rate. The AUC values for each class ranged from 0.91 to 1.00, reflecting high separability between the classes. Most classes, particularly Class 2, Class 3, and Class 7, achieved near-perfect AUC scores, further validating the model's high performance.

XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is a highly efficient ML algorithm based on decision trees that employs gradient boosting techniques. It has a great ability to handle complex structured data sets, making it a strong candidate for classification tasks like malware detection. [17]. Table *5* represents the evaluation metrics of different classes for XGBoost classifier algorithm.

Class	Precision	Recall	F1-Score
Class 1	0.94	0.93	0.94
Class 2	0.99	0.98	0.99
Class 3	1.00	0.99	0.99
Class 4	0.95	0.95	0.95
Class 5	1.00	0.50	0.67
Class 6	0.99	0.91	0.95
Class 7	0.95	1.00	0.97
Class 8	0.97	0.92	0.94
Class 9	0.98	0.98	0.98

TABLE 5: Evaluation Metrics of Different Classes for XGBoost Classifier Algorithm

After training the model, it was evaluated on the test set. The accuracy of the XGBoost model was 0.958, indicating a strong ability to distinguish between various classes of malware. From Table *5*, it is seen that Class 5, which has a very small sample size, exhibited a lower recall, which is 0.50, but the precision remained perfect at 1.00. This indicates the model is accurate when it makes a positive prediction for this class, but it misses half of the actual instances due to the limited number of samples.

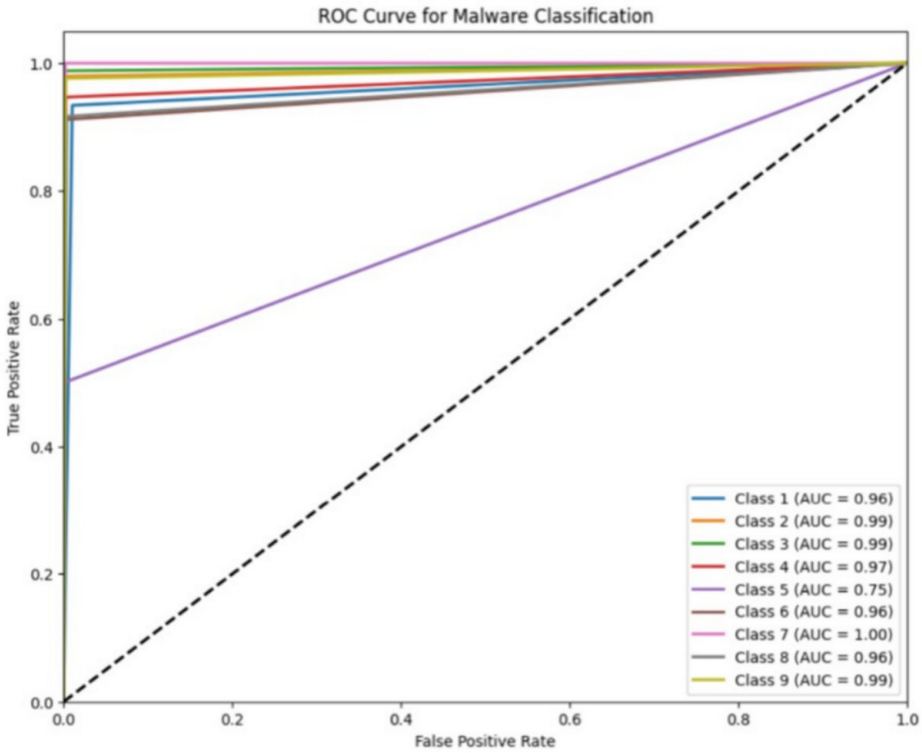

FIGURE 4: Receiver Operating Characteristic (ROC) Curve of the Extreme Gradient Boost Classifier for Each Class

AUC, Area Under the Curve

Figure *4* represents the ROC curve of the Extreme Boost Classifier for each class. To further evaluate the classifier's performance, the ROC curves were plotted for each class. The AUC values were calculated, indicating that most classes performed extremely well with AUC values close to 1.00, showing high separability.

DL models

Artificial Neural Networks

An artificial neural network (ANN) is a computational model inspired by the human brain's structure. It comprises multiple interconnected neurons (nodes) organized into layers, including an input layer, hidden layers, and an output layer. Each neuron processes the input data and applies a non-linear activation function to generate an output, which is then passed to the next layer. In classification tasks such as malware detection, the ANN learns to map input features (malware attributes) to specific class labels (malware types) [18]. In this study, a custom-built ANN was designed and implemented to classify malware samples. The architecture of the ANN included multiple hidden layers, each consisting of several neurons. The model was trained on the encoded feature set using backpropagation and the Adam optimizer to minimize the categorical cross-entropy loss. The custom-built ANN achieved an accuracy of 0.959 on the test data, showcasing its ability to classify different malware types effectively. Table *6* represents the evaluation metrics of different classes for ANN.

Class	Precision	Recall	F1-Score
Class 1	0.92	0.92	0.92
Class 2	0.99	0.98	0.98
Class 3	1.00	0.99	1.00
Class 4	0.95	0.99	0.97
Class 5	0.50	0.17	0.25
Class 6	0.93	0.95	0.94
Class 7	0.96	1.00	0.98
Class 8	0.94	0.92	0.93
Class 9	0.97	0.99	0.98

TABLE 6: Evaluation Metrics of Different Classes for Artificial Neural Network

From Table *6*, it is seen that Class 5 has the lowest precision as 0.50, recall as 0.17, and F1-score as 0.25, which may be attributed to the small number of samples in this class. This class imbalance likely affected the model's ability to generalize for Class 5.

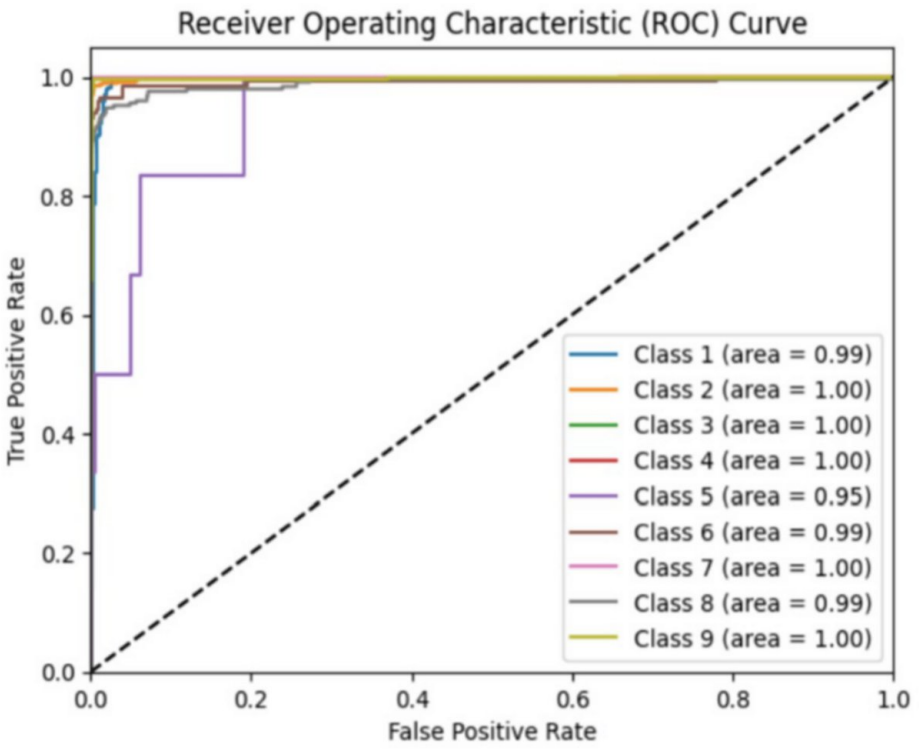

FIGURE 5: Receiver Operating Characteristic Curve of the Artificial Neural Networks for Each Class

Figure *5* represents the ROC curve of ANNs for each class. The AUC for each class ranged from 0.95 to 1.00, showcasing the ANN's strong pattern recognition performance across all malware types. The ROC curves for the 10 malware classes indicated high sensitivity and specificity, with AUC values close to 1.0 for most classes, but for class 5, it has the best AUC value compared to all ML models compared to its strong pattern recognition and its architecture.

Attention-Based Artificial Neural Network

Attention mechanisms have recently gained prominence in DL due to their ability to focus on the most relevant parts of the input data. In the context of malware classification, an attention-based ANN can prioritize certain features over others, enhancing the model's ability to distinguish between different malware classes. By selectively addressing specific features, the model becomes more effective in handling complex patterns and dependencies in the data [19].

The attention-based ANN used in this study incorporates attention layers to enhance feature representation. The model consists of a typical neural network architecture with additional attention mechanisms that assign different importance weights to input features during classification. This allows the model to focus on the most relevant aspects of the input data when making predictions [20].

The attention ANN achieved a higher accuracy of 0.97, surpassing the performance of the custom-built ANN and the traditional ML models used in this study. The model exhibited excellent performance across most classes, particularly in handling complex and imbalanced datasets. Table *7* represents the evaluation metrics of different classes for attention-based ANN.

Class	Precision	Recall	F1-Score
Class 1	0.93	0.96	0.95
Class 2	0.99	0.98	0.99
Class 3	1.00	1.00	1.00
Class 4	0.95	0.97	0.96
Class 5	0.67	0.67	0.67
Class 6	0.95	0.91	0.93
Class 7	0.96	1.00	0.98
Class 8	0.97	0.92	0.95
Class 9	0.96	0.97	0.96

TABLE 7: Evaluation Metrics of Different Classes for Attention-Based Artificial Neural Network

The attention ANN has an evenly balanced error rate for Class 5 as 0.67. The model is neither too conservative, that is, missing a lot of true positives, nor too optimistic, that is, predicting too many false positives. Although 0.67 is considerable improvement over other models, it still leaves room for further enhancement. The relatively low precision and recall suggest that Class 5 remains challenging for the model, likely due to the small dataset size and perhaps inherent difficulty in distinguishing Class 5 features from other classes.

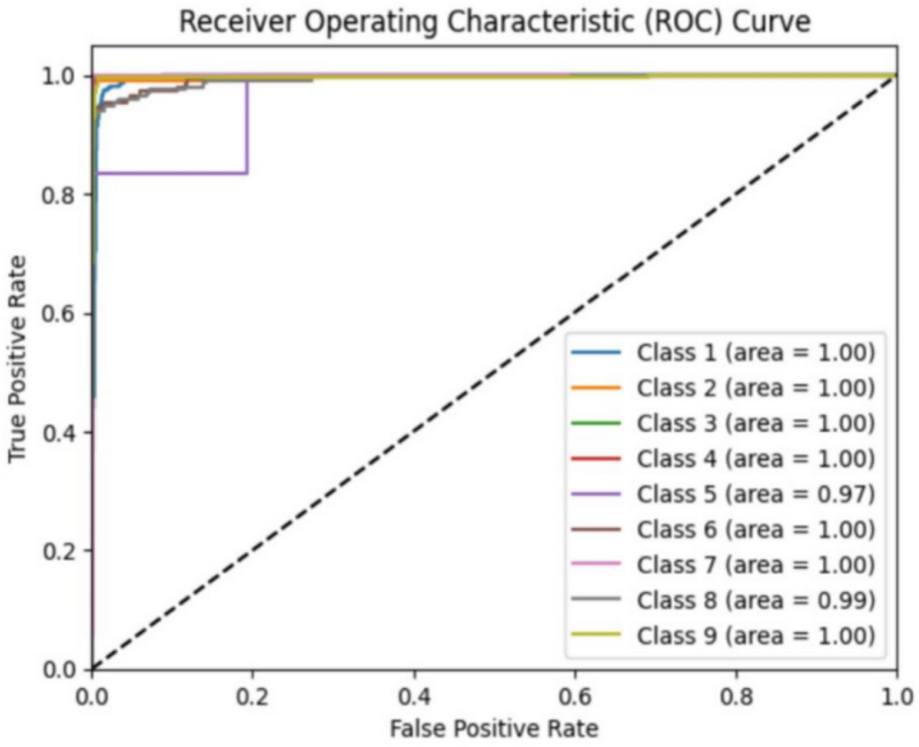

FIGURE 6: Receiver Operating Characteristic Curve of the Attention-Based Artificial Neural Network for Each Class

Figure *6* represents the ROC curve of the attention-based ANN for each class. To further evaluate the classifier's performance, the ROC curves were plotted for each class. The AUC values were calculated, indicating that most classes performed extremely well with AUC values close to 1.00, showing high separability. However, in Class 5, it has performed better than the ANN, which means attention ANN can capture more patterns than a normal ANN.

Evaluation metrics

Accuracy Score

Accuracy score is defined as the ratio of correctly predicted instances (true positives and true negatives) to the total number of instances in the dataset. Mathematically, accuracy is expressed in Equation 1.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Number of Observations}} \rightarrow 1$$

Precision Score

Precision is defined as the ratio of correctly predicted positive instances (true positives) to the total number of instances predicted as positive (the sum of true positives and false positives). Mathematically, precision is expressed in Equation 2.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \rightarrow 2$$

Recall Score

Recall is defined as the ratio of correctly predicted positive instances (true positives) to the total number of actual positive instances (the sum of true positives and false negatives). Mathematically, recall is expressed in Equation 3.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \rightarrow 3$$

F1-Score

F1-Score is a harmonic mean of precision and recall, offering a single metric that balances both metrics in situations with uneven class distribution. Mathematically, the F1-score is expressed in Equation 4.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \rightarrow 4$$

Results

Both ML and DL models achieved high classification accuracy, yet DL models displayed a marked edge in performance. The decision tree and random forest classifiers, which provide interpretable flowchart-like decision structures, achieved accuracy of approximately 0.94 and 0.96, respectively. Random forest's ensemble approach, utilizing multiple decision trees, mitigated over-fitting more effectively than decision trees alone and provided more stable predictions. However, both ML models exhibited difficulties with class imbalances, as demonstrated by the low recall and F1-scores for Class 5, which had limited data samples. This weakness suggests that although ML models can perform well on balanced datasets, they may struggle with nuanced or underrepresented data.

The low recall and F1-scores for Class 5 highlight the limitations of traditional ML when handling class imbalances, a common issue in cybersecurity datasets. For infrequent classes, ML models failed to generalize effectively, often misclassify samples due to limited exposure to diverse patterns. In contrast, DL models, particularly those incorporating attention mechanisms, exhibited resilience against these imbalances. Attention-based ANN, for instance, assigned different importance to specific features, effectively learning which attributes to priority for better distinction among classes. Consequently, the attention-based ANN achieved significantly higher recall for challenging classes, indicating a deeper capacity for learning subtle patterns within the malware dataset. Custom ANN and attention-based ANN significantly outperformed traditional ML models, achieving accuracies of 0.96 and 0.97, respectively.

The ANN models' multi-layer architecture allowed them to recognize complex, high-dimensional patterns in the data, and the attention-enhanced model provided even better performance. Attention mechanisms empowered the ANN to identify and focus on the most relevant features within each sample, a notable advantage for detecting variations across different malware classes. This increased adaptability to complex, and often imbalanced data patterns made the attention-based ANN particularly effective for this study. The model's balanced performance across classes illustrates how attention mechanisms allow DL models to make more accurate predictions with a reduced false positive and false negative rates, a crucial

factor in cybersecurity applications.

Moreover, all experiments were performed on a system equipped with an NVIDIA Tesla T4 GPU and 32 GB RAM, providing a reference for the computational requirements needed to train DL models within reasonable timeframes. While DL models are typically more resource-intensive, the attention-based ANN demonstrated scalable performance, maintaining high accuracy even as dataset size and dimensionality increased. Additionally, to approximate real-world applicability, the dataset was partitioned to simulate unknown malware variants in the test set, and the model maintained robust performance, suggesting generalizability beyond the training distribution.

ROC-AUC analysis across all models showed strong sensitivity and specificity, but DL models, especially attention-based ANN, consistently maintained higher AUC values across most classes. For the challenging Class 5, the attention based ANN performed notably better than other models, showcasing its ability to discern complex, overlapping patterns that simpler models might overlook. The AUC close to 1.0 for most classes confirmed the model's high separability, underscoring its robustness in handling the subtle patterns required for effective malware classification. Table *8* represents the comparison of evaluation metrics for different models.

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree Classifier	0.947	0.95	0.95	0.95
Random Forest	0.959	0.99	0.96	0.97
XGBoost	0.958	0.98	0.96	0.97
Artificial Neural Network	0.959	0.96	0.96	0.97
Attention-Based Artificial Neural Network	0.971	0.97	0.97	0.97

TABLE 8: Comparison of Evaluation Metrics for Different Models

Discussion

This study demonstrates that DL models, particularly those that incorporate attention mechanisms, provide better performance over traditional ML models for malware classification. The sophisticated architecture of DL models enables them to better recognize hierarchical and intricate patterns in complex, high-dimensional datasets. This capability allows them to tackle challenges like class imbalance and subtle feature differences, where ML models might fall short. The findings suggest that attention-enhanced DL models, with their improved accuracy and resilience against imbalances, hold promise for advancing malware detection in cybersecurity.

Conclusions

In conclusion, the application of DL models, especially attention-based ANNs, has shown exceptional performance in malware classification. These models offer significant advantages in terms of accuracy, robustness, and the ability to handle complex patterns. Their capacity to capture subtle distinctions between malware classes highlights their importance in cybersecurity applications. For future work, exploring more advanced architectures, such as transformer-based models, could further enhance detection accuracy. Additionally, addressing data imbalance through synthetic data generation techniques or leveraging transfer learning may improve performance, particularly for underrepresented classes like Class 5. Integrating real-time detection capabilities may further expand the practical impact of these models in dynamic cybersecurity environments.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Thenmozhi M, Ravin D, Akshwin T

Critical review of the manuscript for important intellectual content: Thenmozhi M, Ravin D, Akshwin T

Supervision: Thenmozhi M

Acquisition, analysis, or interpretation of data: Ravin D, Akshwin T

Drafting of the manuscript: Ravin D, Akshwin T

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Aslan ÖA, Samet R: A comprehensive review on malware detection approaches. IEEE Access. 2020, 8:6249-6271. 10.1109/access.2019.2963724
2. Rathore H, Agarwal S, Sahay SK, Sewak M: Malware detection using machine learning and deep learning. Big Data Analytics. Mondal A, Gupta H, Srivastava J, Reddy P, Somayajulu D (ed): Springer, Cham; 2018. 11297:402-411. 10.1007/978-3-030-04780-1_28
3. Bugra C, Erdogan D: Malware classification using deep learning methods. ACMSE '18: Proceedings of the 2018 ACM Southeast Conference. 2018, 1-5. 10.1145/3190645.3190692
4. Pascanu R, Stokes JW, Sanossian H, Marinescu M, Thomas A: Malware classification with recurrent networks. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia. 2015, 1916-1920. 10.1109/ICASSP.2015.7178304
5. Kalash M, Rochan M, Mohammed N, Bruce NDB, Wang Y, Iqbal F: Malware classification with deep convolutional neural networks. 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France. 2018, 1-5. 10.1109/NTMS.2018.8328749
6. Nari S, Ghorbani AA: Automated malware classification based on network behavior. 2013 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, USA. 2013, 642-647. 10.1109/ICCNC.2013.6504162
7. Aslan Ö, Yilmaz AA: A new malware classification framework based on deep learning algorithms. IEEE Access. 2021, 9:87936-87951. 10.1109/access.2021.3089586
8. Xue D, Li J, Lv T, Wu W, Wang J: Malware classification using probability scoring and machine learning. IEEE Access. 2019, 7:91641-91656. 10.1109/access.2019.2927552
9. Al-Janabi M, Altamimi AM: A comparative analysis of machine learning techniques for classification and detection of malware. 2020 21st International Arab Conference on Information Technology (ACIT), Giza, Egypt, 2020. 2020, 1-9. 10.1109/ACIT50332.2020.9300081
10. Yadav P, Menon N, Ravi V, Vishvanathan S, Pham TD: EfficientNet convolutional neural networks-based Android malware detection. Computers & Security. 2022, 115:102622. 10.1016/j.cose.2022.102622
11. Ravi V, Alazab M, Selvaganapathy S, Chaganti R: A multi-view attention-based deep learning framework for malware detection in smart healthcare systems. Computer Communications. 2022, 195:73-81. 10.1016/j.comcom.2022.08.015
12. Ganesan S, Ravi V, Krichen M, SV, Alroobaea R, Soman KP: Robust malware detection using residual attention network. 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA. 2021, 1-6. 10.1109/ICCE50685.2021.9427623
13. Panconesi A, Marian M, Cukierski W, WWW BIG - Cup Committee. Microsoft Malware Classification Challenge (BIG 2015). Kaggle. (2015). https://www.kaggle.com/competitions/malware-classification.
14. Wang Y, Yao H, Zhao S: Auto-encoder based dimensionality reduction. Neurocomputing. 2016, 184:232-242. 10.1016/j.neucom.2015.08.104
15. Swain PH, Hauska H: The decision tree classifier: Design and potential. IEEE Transactions on Geoscience and Remote Sensing. 1977, 15:142-147. 10.1109/tge.1977.6498972
16. Breiman L: Random forests. Machine Learning. 2001, 45:5-32. 10.1023/a:1010933404324
17. Chen T, Guestrin C: XGBoost: A scalable tree boosting system. KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016, 785-794. 10.1145/2939672.2939785
18. Zou J, Han Y, So SS: Overview of artificial neural networks. Methods in Molecular Biology™. Livingstone DJ (ed): Humana Press, Clifton, NJ; 2008. 458:14-22. 10.1007/978-1-60327-101-1_2
19. Vaswani A, Shazeer N, Parmar N, et al.: Attention is all you need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.. 2017, 1-11.
20. Manca V: Artificial neural network learning, attention, and memory. Information. 2024, 15:387. 10.3390/info15070387