ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Conference Proceedings

Open Access

Evaluating and Comparing the Efficiency of Malware Identification for Cyber Security Using Machine Learning Algorithms (Approved by ICITET 2024)

Purna Chandra Rao Chinta[1*], Chethan Sriharsha Moore[2], Manikanth Sakuru[3], KishanKumar Routhu ADP[4], Varun Bodepudi[5] and Gagan Kumar Patra[6]

[1]Microsoft, Support Escalation Engineer, USA

[2]Microsoft, Support Escalation Engineer, USA

[3]JP Morgan Chase, Lead Software Engineer, USA

[4]Senior Solution Architect, USA

[5]Deloitte Consulting LLP, Senior Solution Specialist, USA

ABSTRACT

Malware detection is a critical task in information security, and it should be able to detect the occurrence of potential malicious code in memory. Detection methods typical of the past could sometimes not detect the new malware variants because it is highly inaccurate. In this study, it presents a robust malware detection model employing Long Short-Term Memory (LSTM) networks to deal with class imbalance and feature redundancy. Within a well-structured preprocessing pipeline, it uses feature selection through correlation analysis and class balancing with SMOTE. Experimental results demonstrate that the model performs quite accurately, has an F1 score of 97.97%, accuracy of 99.17%, precision of 98.80%, and recall of 99.17%, proving its reliability and robustness when classifying both benign and Malware instances. Comparative analysis with traditional models, such as Decision Tree and the suggested LSTM model, beats these techniques with the maximum accuracy of 99.09%, according to K-Nearest Neighbors (KNN). The results indicate that the model outperforms malware detection, has good generalization ability and low overfitting, and therefore is a very good solution for real-world cybersecurity applications.

***Corresponding author**

Purna Chandra Rao Chinta, Microsoft, Support Escalation Engineer, USA.

Received: December 16, 2024; **Accepted:** December 20, 2024; **Published:** December 28, 2024

Keywords: Cybersecurity, Cyber Threats, Malware, Malware Detection, Classification, Machine Learning (ML), Malware Dataset

Introduction

In the age of modern digital times, technological advancement has entirely changed the operation of people, businesses and governments. As the digital world expands more and more on cloud computing, mobile platforms, and IoT devices, the digital world is speeding up with increasing reliance on interconnected systems. Still, as with these innovations, there is a growing surface for cybercriminals to exploit to compromise systems and steal sensitive information. In recent years, cyber threats have changed a lot and are now more complex, targeted, and persistent [1]. Among the threats introduced by attackers, malware is still one of the most dangerous tools used. Infected with malware, networks can be infiltrated, operations disrupted, data exfiltrated, and in the worst cases paralyzed entire infrastructures [2]. Two, since it is capable of obfuscating, polymorphing, as well as stealthily propagating itself, which makes it very dangerous for cybersecurity [3].

Analysis and detection of malware have now become the need of the hour as a result of malware assaults' growing complexity and volume. Although heuristic and signature-based strategies are quite successful in recognizing known threats, they do not do well when these present themselves in the form of newly emerging threats or highly modified malware [4,5]. With this, the more sophisticated detection techniques that can be adapted to the dynamic threat environment have changed. The other portions of malware are Malware detection and classification, which groups malware into families according to how they behave, structure and origin [6,7]. Such classification helps in faster response, Improved comprehension of danger actors and the deployment of appropriate preventative measures [8]. These are key elements of any full-blown cybersecurity scheme.

As researchers and practitioners continue seeking the ability to increase the current malware detection system's precision, speed, and flexibility, they are more inclined towards using ML algorithms. The ML capability to automatically learn patterns from a large dataset, detect anomalies, and generalize from past experience in

order to identify otherwise unseen threats [9,10]. ML has been found promising for improving the detection rates and reducing false positives if applied through Static code analysis, dynamic study of execution behaviour, and hybrid approaches.

Motivation and Contribution of the Paper

This research is motivated by the increase in malware attacks, which are becoming more sophisticated and frequent, and thus pose a serious cybersecurity threat. Because traditional detection methods tend to fail in dealing with new and evolving malware variants, it is desirable to develop a general scheme for the detection of malicious code. This study endeavors to improve the detection system by means of a more intelligent and adaptive detection system by taking advantage of the sequential learning capabilities of the LSTM networks. Data balancing, feature selection, and DL are integrated to form a fully integrated approach that preserves the model's effectiveness and generalizability in practical settings in addition to improving detection accuracy. The contribution to the field that this paper makes is in several ways:

- Preprocessed the malware data from missing values and removed irrelevant features to ensure malware data quality.
- To overcome class unbalance, use SMOTE to improve minority class detection.
- The model reduces dimensionality using a correlation-based feature selection method to increase efficiency as well as interpretability without reducing accuracy.
- The paper proposes to use an LSTM architecture for malware detection as it is effective in detecting sequential patterns, which could indicate malicious activity in system behavior or data flow.
- A well-rounded assessment of the model's efficacy is evaluated using several performance metrics, including F1 score, recall, accuracy, and precision.

Justification and Novelty of paper

This study is justified by the shortage of traditional malware detection techniques, which are unsuitable in various cases due to their inability to adapt to new and sophisticated attack patterns, especially with imbalanced datasets. The paper introduces a novel end-to-end framework that consists of LSTM networks joined with a carefully designed preprocessing pipeline to tackle the deficiencies. It involves the handling of missing data, the reduction of features using correlation analysis, and balancing the classes with SMOTE. This work contributes to the combination of these approaches, which improves both detection accuracy and robustness while keeping model efficiency when deploying these models in the real-world malware.

Structure of Paper

The structured of this paper is organized as follows: Sections II provide a background study on malware detection in the context of cybersecurity. The study's methodology is described in Section III. The results are shown in Section IV, together with a thorough analysis and discussion. Section V, which wraps up the study, offers recommendations for future research directions.

Literature Review

In the current study, several significant research efforts related to malware detection have been reviewed and analyzed. Table I presents the background study on malware detection for cybersecurity, highlighting the datasets used, ML models applied, their performance, and overall contributions.

Han, et al. aspects represent the network, registry, and file activities, the structural elements and underlying activities that interface with the operating system, respectively. To verify the efficacy of MalInsight, a comprehensive test is carried out on an actual malware dataset. Their test findings demonstrate that MedInsight can identify new and undetected malware with a 97.21% accuracy rate and instances of obfuscated malware with a 99.76% accuracy rate. Furthermore, MedInsight has a 94.2% accuracy rate in classifying malware samples into their families, which is almost 9% better than the conventional detection approach that uses the API sequence as the dynamic behavior characteristics [11].

Noor, et al. provide a framework for automating the attribution of cyberthreats. From May 2012 to February 2018, 327 CTI reports that were taken from incident reports that were made public were used to train and test five ML classifiers that leverage these factors. In contrast to other publicly accessible CTA profiles, which have a 33% accuracy rate, it is seen that the CTA profiles obtained have an 83% accuracy rate in identifying cyber hazards. Additionally, compared to earlier classifiers, the DLNN-based classifier assesses cyber threats more accurately (94%) [12].

Navarro, et al. provide an ML method to examine the intricate network that results from modelling the interactions between application and system components using an ontology-based framework. It illustrates the ontological model for the 4570 applications in the Android ecosystem in question operate on a network with over 120,000 edges and 55,000 nodes. According to experiments, a classifier operating on top of this intricate depiction can recognize and determine 24 features that relate to 70 critical network nodes associated with malware activity, achieving an impressive security feat with an accuracy of 88% and precision of 91% [13].

Cohen and Nissim provide a special method for the accurate identification of ransomware on virtual servers housed in a business's private cloud. conducted a reliable analysis of volatile memory dumps from a virtual system (memory forensics) using the Volatility framework to provide wide descriptive meta-features. Five lengthy studies of various levels of difficulty extensively evaluated their method on two different well-known servers (an email server and an IIS server). The RF classifier is used, the findings demonstrate that their technique can identify both known and unknown ransomware as well as anomalous states aof a virtual computer, generating the findings were TPR = 1, FPR = 0.052, F-measure = 0.976, and AUC = 0.966 [14].

Milosevic, Dehghantanha and Choo give two methods for static analysis of Android malware that are aided by ML. A bag-of-words representation model is used in one approach to assess source code, whereas permissions are used in another. The OWASP Seraphim droid Android app, available on the Google Play Store, incorporates their computationally inexpensive permission-based technique. Their results demonstrate that The source code-based categorization model has an F-score of 95.1%, whereas the permission-based model has an F-measure of 89% [15].

Feizollah, et al. investigate the usefulness of both Android intents, both explicit and implicit, as a defining characteristic for identifying phoney applications. Additionally, it makes the case why this kind of functionality is not the best option. It ought to be combined with other well-known traits. It experimented with a dataset of 7406 apps, of which 1846 were clean and 5560 were infected. The results show a 91% detection rate with Android intent and an 83% detection rate with Android permission. Furthermore, testing both traits together yields a 95.5% detection rate [16].

Table 1: Overview of Recent Studies on Malware Detection for Cyber Security using Machine Learning Algorithms

Author	Proposed Work	Dataset	Key Findings	Challenges/Gaps
Han et al.	MalInsight for detecting and classifying malware using structural features and OS interactions	Real-world malware dataset	Detection accuracy of 99.76%, new/unseen malware detection 97.21%, family classification accuracy 94.2%	Needs real-time deployment validation; limited insight into performance across various malware types
Noor et al.	Cyber threat attribution based on NLP and CTA profiles	327 CTI reports (May 2012 – Feb 2018)	Precision of 83%, DLNN classifier accuracy of 94%	Dependence on quality and consistency of CTI reports
Navarro et al.	Ontology-based Android malware detection framework	4570 Android apps (graph model: 55K nodes, 120K edges)	Accuracy of 88%, precision of 91%; 24 key features identified	High complexity and computational requirements due to graph-based analysis
Cohen and Nissim	Ransomware detection on virtual servers using memory forensics	dumps of IIS and email servers' volatile memory	TPR = 1, FPR = 0.052, AUC = 0.966 (using Random Forest), and F-measure = 0.976	Limited to virtual environments; performance in diverse infrastructures not shown
Milosevic, Dehghantanha and Choo	Static Android malware detection via permissions and source code (BoW)	Not explicitly stated; integrated into OWASP Seraphim droid	F-measure of 89% (permissions) and F-score of 95.1% (source code)	Static analysis limits ability to detect obfuscated or dynamic behaviors
Feizollah et al.	Use of Android Intents for malware detection	7406 apps (1846 clean, 5560 infected)	Detection rate: 91% (Intents), 83% (permissions), 95.5% (combined)	Intent-based features alone not sufficient; best used in combination

Research Methodology

A systematic strategy to developing a reliable malware detection model with LSTM is part of the study process. The study begins with data collection from a Kaggle dataset including both benign and malware samples. The data undergoes preprocessing, including handling missing values using mean and mode imputation, removing irrelevant features, and label encoding the target class into binary values (0 for benign, 1 for malware). To address class imbalance, the minority (malware) class is artificially oversampled using the SMOTE approach. A correlation-based strategy is utilized for feature selection to improve model performance and decrease dimensionality. 20% of the dataset is used for testing, while the remaining 80% is used for training. Then, using 64 units and a dropout rate of 0.3 to prevent overfitting, an LSTM model is trained over 30 epochs using the Adam optimizer. Measures like F1-score, recall, accuracy, and precision that are derived from the confusion matrix are used to assess the model. Figure 1 provides the entire process and procedures.

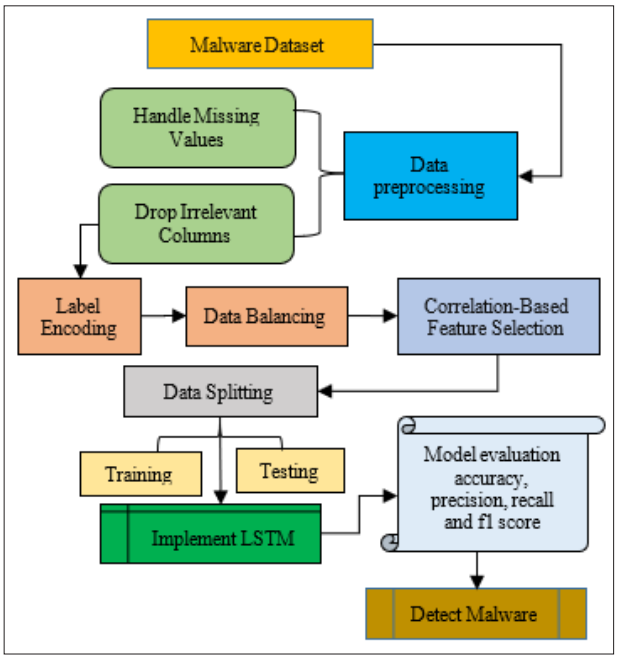

Figure 1: Proposed flowchart for Malware Detection

Each step of a proposed flowchart for Malware Detection for Cyber Security is provided below:

Data Collection

The Kaggle website offers the malware dataset. It consists of 215 unique characteristics collected from more than 15,000 Android apps, of which 9476 are malicious and 5560 are benign. The dataset was produced with keeping in mind malware classification and detection. Figure 2 shows how the malware and benign classes are distributed throughout this dataset.

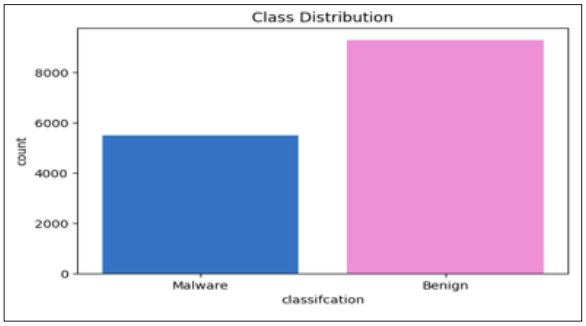

Figure 2: Bar Graph for Class Distribution

A bar graph displaying the distribution of classes between the malware and benign categories is shown in Figure 2. The blue bar representing "Malware" has a count of approximately 5500, while the pink bar for "Benign" shows a significantly higher count, exceeding 9000. This visual representation indicates an imbalanced dataset where the quantity of malware samples is significantly smaller than the quantity of benign ones.

Data Preprocessing

The benchmarked Malware Dataset undergoes comprehensive preprocessing to guarantee the dependability and quality of data for analysis [17]. Given the prevalence of missing values, errors, and duplications in raw data, this phase is crucial. The following pre-processing steps are listed below:

- **Handle Missing Value:** The central tendency is maintained by filling up missing values with the column mean in numerical columns. Numerical columns with missing values are filled up using their mean values in order to preserve central tendency; to maintain the original distribution, categorical columns are imputed using the mode.
- **Dropping Irrelevant Columns:** Some columns, like hash, serve as unique identifiers and do not contribute useful information for classification tasks. These columns are eliminated from the dataset to simplify the training features and enhance model efficiency.

Label Encoding of Target Class

Label encoding of the target class involves transforming the categorical values in the classification column into numerical form to make them suitable for ML models demand numerical inputs for their operation. In this process, the two categories present 'Benign' for benign and 'Malware' for malware are mapped to numerical labels, with 'Benign' encoded as 0 and 'Malware' as 1. This binary encoding helps the model efficiently identify patterns linked to each class by the instructors and evaluators must maintain consistent assessment of malicious and non-malicious samples during the entire educational process.

Data Balancing

SMOTE is frequently used to address issues of class inequality. This resampling method makes use of artificial data points that are produced by interpolating new instances between the minority class's available data points. The Smote method will work effectively.

Correlation-Based Feature Selection

Feature selection helps reduce computation time by identifying the most important aspects of the analyzed topic. Nonetheless, a correlation-based strategy is suggested in their work to minimize excessive dimensionality, cut down on computation time, and choose the optimal feature combinations to improve training and assessment process performance.

Data Splitting

The dataset is separated into two parts: 20% is reserved for performance evaluation, while the other 80% is utilized to train the model.

Proposed LSTM Models

The vanishing gradient issue that classical RNN had was explicitly addressed by LSTM networks. This problem limits the RNN's capacity to recall and transmit useful information over a lengthy period, which makes it more difficult to effectively capture temporal correlations [18]. The following are the essential elements of an LSTM cell: Equations (1) through (4).

Cell State (u_t): The LSTM can progressively retain important information because to the cell state, which serves as the memory component. The network can detect long-term dependencies in the data by employing specific gates to selectively regulate what data is saved or discarded.

$$u_t = \tanh\left(W_c * \left[h_{(t-1)}, x_t\right] + b_c\right) \quad (1)$$

Input Gate (i_t): The input gate controls how fresh data enters the cell state. It chooses which values to include in the cell state depending on the current input and the prior hidden state.

$$i_t = sigmoid\ (W_i * \left[h_{(t-1)}, x_t\right] + b_i \quad (2)$$

Forget Gate (f_t): The forget gate specifies which cell state information should be removed. It enables the LSTM to remove unneeded or obsolete data from prior time steps.

$$f_t = sigmoid\ \left(W_f * \left[h_{(t-1)}, x_t\right] + b_f\right) \quad (3)$$

Output Gate (O_t): The output gate filters the data on cell statuses to identify the current hidden state. It governs the information provided to the next time step.

$$O_t = sigmoid\ \left(W_o * \left[h_{(t-1)}, x_t\right] + b_o\right) \quad (4)$$

These gates increase efficiency by using their decision-making ability to decide which data should be added to the cell state and which should be discarded. 64 units per layer, 32 batch sizes, and a learning rate of 0.001 optimized using Adam are all used in the suggested LSTM model. A dropout rate of 0.3 is used to prevent overfitting. The model's loss function is Binary Cross-Entropy, its output is Sigmoid, and its hidden layers are ReLU. Trained for 30 epochs with a sequence length of 100, the model effectively captures temporal dependencies, enhancing malware detection.

Evaluation Metrics

In ML, a confusion matrix is used to evaluate the performance of a classification model. The quantity of TP, FP, TN, and FN in each class is called a confusion matrix. The number of correct forecasts for the positive class is TP, The positive class has FP

wrong predictions, the negative class has TN accurate predictions, and the negative class has FN incorrect guesses. Accuracy, precision, recall, and other assessment metrics may be computed using a confusion matrix. Accuracy, precision, and recall are the assessment measures utilized in this study to assess and contrast various models.

Accuracy: The proportion of accurate predictions to all of the testing dataset predictions are known as accuracy. It is shown as Equation (5)-

$$Accuracy = \frac{TP+TN}{TP+Fp+TN+FN} \quad (5)$$

Precision: It's the proportion of positive class forecasts that turn out to be true predictions. The number of exactly predicted positive observations divided by the total number of planned positive observations yields the interpretation. It is expressed as Equation (6)-

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

Recall: The total number of correct positive forecasts correct positive samples is used to compute recall. If it is given in mathematical form, it is given as Equation (7)-

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

F1 Score: The F1 score is calculated by taking the harmonic mean of accuracy and recall, and how accurate a model is when predicting the classes in a dataset. Mathematically, it is given as Equation (8)-

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

Taken as a set of these metrics together, It is known how accurate the model is and how well it can forecast the target variable.

Results and Discussion

The experimental findings from the suggested malware detection method are examined in this section. Initially, a computer with 16 cores, 38 GB of RAM, and Ubuntu 20.04.3 was used for all testing. According to Table II, the suggested model performs exceptionally well on malware detection tests across all major assessment metrics. The model is such that its accuracy was 99.17% which can be said to be so high in correct classification. It achieves 98.80 percentage precision at identifying true malware instances with only a few false-positive results. The model's 99.17% recall value shows that it can identify nearly all actual malware instances, and its 97.97% F1-score demonstrates a well-balanced trade-off between recall and accuracy. These metrics collectively highlight the robustness and reliability of the GWOA model in detecting malware accurately.

Table 2: Experiment Results Of Proposed Models For Malware Detection On Malware Data

Performance Matrix	LSTM
Accuracy	99.17
Precision	98.80
Recall	99.17
F1-score	97.97

Figure 3 depicts a confusion matrix used to assess the accuracy of a categorization system for detecting malware and benign samples. The matrix is divided into four sections: TN (4947, 65.96%) in the top-left, representing correctly identified benign cases; FP (53, 0.71%) in the top-right, indicating benign cases misclassified as malware; FN (15, 0.20%) in the bottom-left, showing malware cases misclassified as benign; and TP (2485, 33.13%) in the bottom-right, representing correctly identified malware instances. The model demonstrates high accuracy in both detecting benign and malware samples, with minimal misclassification rates.

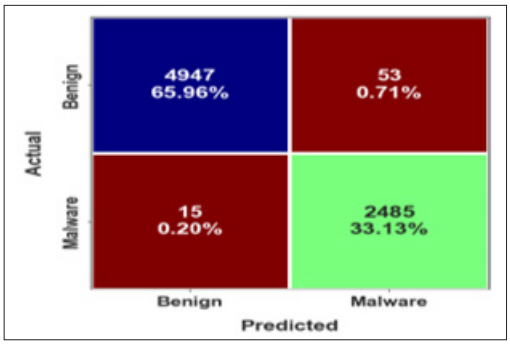

Figure 3: Confusion Matrix of LSTM

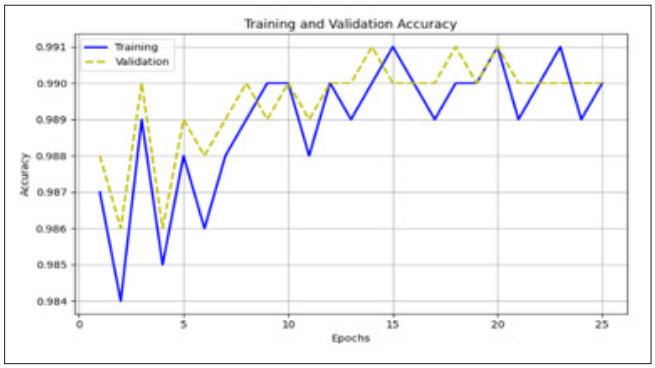

Figure 4: Training and Validation Accuracy of LSTM

Figure 4 depicts the performance of an LSTM model over 25 training epochs, showing the precision on the validation dataset (yellow dotted line) and training dataset (blue solid line). Both training and validation accuracy generally trend upwards, indicating learning. However, the validation accuracy appears to plateau and fluctuates more than the training accuracy after a certain point, suggesting that although. The model's performance on training data keeps getting better, while its ability to generalize to fresh data has been consistent.

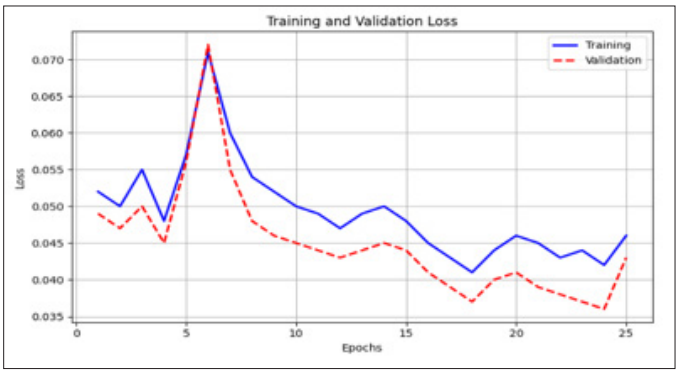

Figure 5: Training and Validation Loss of LSTM

The performance of an LSTM model across 25 epochs is displayed in the graph in Figure 5. Learning is indicated by training loss (blue line) steadily declining while validation loss (red dashed line) drops initially but rises after epoch 7, suggesting potential overfitting as the model starts to lose generalization ability.

Comparison with Discussion

In this section, the experimental results are evaluated against other malware detection models. A comparison of several malware detection methods utilizing a malware dataset is presented in Table III. The accuracy of each model was used to assess its performance. The accuracy of the DT model was 92.2%, while the accuracy of the KNN model was slightly higher, at 94%. This is notably the case of the LSTM model, which outperformed all with 99.09% accuracy, in terms of accuracy, in accurately detecting malware.

Table 3: Comparative Analysis Between Various Models For Malware Detection Using The Malware Da-taset

Models	Accuracy
Decision Tree [19]	92.2
KNN [20]	94
LSTM	99.09

According to the proposed LSTM model, this model offers a high detection accuracy, and a 99.09% detection accuracy for malware detection. Use of the LSTM allows the model to efficiently pick the optimal features, hence eliminating redundancy, thus enhancing the overall performance. This ensures the balance between exploring and exploitation, avoiding overfitting and expediting the model training caused by its fast convergence speed. This implies that the LSTM model is not only superior to traditional methods but also has strong generalization capabilities to unseen data, and thus it is a robust and reliable solution to malware detection.

Conclusion and Future Study

The detection of malware exists as a complicated challenge. The signature-based detection method that is commonly used fails to detect both unknown and zero-day malware. ML methods that use traditional models help identify new harmful programs, yet need advanced expertise to design their characteristics. The research explores LSTM networks for malware detection that tackles standard problems of class imbalance and feature duplication to excel at malicious software detection. The experimental findings indicate that the LSTM model accomplished 99.17% accuracy, together with 98.80% precision and 99.17% recall, and 97.97% F1-score during its task of classifying benign and malware instances. The confusion matrix demonstrates exceptional robustness in the model since it shows low levels of misclassification among benign and malware samples at 0.71% FP and 0.20% FN. Real-world cybersecurity applications find merit in using this LSTM model because it shows both high detection accuracy and effective generalization abilities. The powerful performance of the LSTM model presents an ongoing challenge because it could cause overfitting problems, specifically when dealing with limited data variation. It is advised to further research in terms of improving generalization by using better regularization approaches and increasing the size of the dataset while designing hybrid models that combine LSTM with other ML methods. These activities shall enhance the strength in the detection of malware [19-26].

References

1. Kolluri V (2016) A Pioneering Approach To Forensic Insights: Utilization AI for Cybersecurity Incident Investigations. Int J Res Anal Rev 3.
2. Ucci D, Aniello L, Baldoni R (2019) Survey of machine learning techniques for malware analysis. Comput. Secur 81: 123-147.
3. Bakdash JZ (2018) Malware in the future? Forecasting of analyst detection of cyber events. J Cybersecurity 4.
4. Kozik R (2018) Distributing extreme learning machines with Apache Spark for NetFlow-based malware activity detection. Pattern Recognit Lett 101: 14-20.
5. Sethi K, Kumar R, Sethi L, Bera P, Patra PK (2019) A novel machine learning based malware detection and classification framework. International Conference on Cyber Security and Protection of Digital Services, Cyber Security.
6. Nissim N, Lahav O, Cohen A, Elovici Y, Rokach L (2019) Volatile memory analysis using the MinHash method for efficient and secured detection of malware in private cloud. Comput. Secur 87: 101590.
7. Rosli NA, Yassin W, Rahayu S (2019) Clustering Analysis for Malware Behavior Detection using Registry Data," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 12, 2019, doi: 10.14569/IJACSA.2019.0101213.
8. Ajay Kumara MA, Jaidhar CD (2018) Automated multi-level malware detection system based on reconstructed semantic view of executables using machine learning techniques at VMM. Futur Gener Comput Syst 79: 31-446.
9. Zhong W, Gu F (2019) A multi-level deep learning system for malware detection. Expert Syst Appl 133: 151-162.
10. Karbab EB, Debbabi M (2019) MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports. Digit Investig 28: S77-S87.
11. Han W, Xue J, Wang Y, Liu Z, Kong Z (2019) MalInsight: A systematic profiling based malware detection framework. J Netw Comput Appl 125: 236-250.
12. Noor U, Anwar Z, Amjad T, Choo KKR (2019) A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise. Futur Gener Comput Syst 96: 227-242.
13. Navarro LC, Navarro AKW, Grégio A, Rocha A, Dahab R (2018) Leveraging ontologies and machine-learning techniques for malware analysis into Android permissions ecosystems," Comput Secur 78: 429-453.
14. Cohen A, Nissim N (2018) Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. Expert Syst Appl 102: 158-178.
15. Milosevic N, Dehghantanha A, Choo KKR (2017) Machine learning aided Android malware classification. Comput Electr Eng 61: 266-274.
16. Feizollah A, Anuar NB, Salleh R, Suarez-Tangil G, Furnell

S (2017) AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection. Comput Secur 65: 121-134.

17. Immadisetty A (2016) Edge Analytics vs. Cloud Analytics: Tradeoffs in Real-Time Data Processing. J Recent Trends Comput Sci Eng 13: 42-52.
18. Tarafdar R, Han Y (2018) Finding Majority for Integer Elements. J Comput Sci Coll 33: 187-191.
19. Sharma MS (2017) Real-Time Malware Detection Using Machine Learning Algorithms. Redshine Arch 2: 12-16.
20. Selamat NS, Ali FHM (2019) Comparison of malware detection techniques using machine learning algorithm. Indones J Electr Eng Comput Sci 16: 435.
21. Kuraku S, Kalla D, Samaah F, Smith N (2023) Cultivating proactive cybersecurity culture among IT professional to combat evolving threats. International Journal of Electrical, Electronics and Computers 8.
22. Kalla D, Smith N, Samaah F, Polimetla K (2022) Enhancing Early Diagnosis: Machine Learning Applications in Diabetes Prediction. Journal of Artificial Intelligence & Cloud Computing 2-7.
23. Kuraku DS, Kalla D (2023) Impact of phishing on users with different online browsing hours and spending habits. International Journal of Advanced Research in Computer and Communication Engineering 12.
24. Kalla D, Kuraku S (2023) Phishing website url's detection using nlp and machine learning techniques. Journal of Artificial Intelligence 5: 145.
25. Kuraku DS, Kalla D, Samaah F (2022) Navigating the link between internet user attitudes and cybersecurity awareness in the era of phishing challenges. International Advanced Research Journal in Science, Engineering and Technology 9.
26. Kuraku DS, Kalla D, Smith N, Samaah F (2023) Exploring How User Behavior Shapes Cybersecurity Awareness in the Face of Phishing Attacks. International Journal of Computer Trends and Technology 71.