

Prediction of novel malware using hybrid convolution neural network and long short-term memory approach

Nagababu Pachhala¹, Subbaiyan Jothilakshmi¹, Bhanu Prakash Battula²

¹Department of Information Technology, Faculty of Engineering and Technology, Annamalai University, Annamalai nagar, India

²Department of Computer Science and Engineering, KKR & KSR Institute of Technology and Sciences, Guntur, India

Article Info

Article history:

Received Oct 25, 2023

Revised Feb 22, 2024

Accepted Mar 18, 2024

Keywords:

Bag-of-words

Convolution neural network

Long short-term memory

Deep learning

Malware

Malware prediction

ABSTRACT

The rapid evolution of network communication technologies has led to the emergence of new forms of malware and cybercrimes, posing significant threats to user safety, network infrastructure integrity, and data privacy. Despite efforts to develop advanced algorithms for detecting malicious activity, constructing models that are both accurate and reliable remains a challenge, especially in handling vast and dynamically shifting data patterns. The prevalent bag-of-words (BOW) method, while widely used, falls short in capturing crucial spatial and sequence information vital for detecting malware patterns. To address this challenge, the work presented in this paper proposes hybrid convolution neural network-long short-term memory network (CNN-LSTM) combination models, leveraging CNN's spatial information extraction and LSTM's temporal modeling capabilities. Focused on predicting the infiltration of malicious software into personal computers, the proposed hybrid CNN-LSTM model considers factors such as location, firmware version, operating system, and anti-virus software. The proposed models undergo training and evaluation using Microsoft's malware dataset, demonstrating superior performance compared to traditional CNN and LSTM models. The CNN-LSTM model achieves an impressive accuracy of 95% on the Microsoft malware dataset, highlighting its effectiveness in malware detection.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Nagababu Pachhala

Department of Information Technology, Faculty of Engineering and Technology, Annamalai University

Annamalainagar - 608002, Tamil Nadu, India

Email: nagababupachhala2024@gmail.com

1. INTRODUCTION

The term "malicious software [1]" which is shortened as "malware", concerns to any program that is conceived with the target of causing turmoil on a processor system or network [2]–[4]. It can take many forms, such as viruses, trojans, and ransomware, and may be disseminated via a variety of channels, including malicious websites, email attachments, and even social media even through software vulnerabilities [5]–[8]. The traditional approach to detecting malware has been signature scanning, which is a technique that looks for known patterns of malicious code. This method compares the code of a program against [9] a database of known signs of malicious software and in the event that a match is identified, the program is identified as malicious software [10]. However, as malware has evolved to become more sophisticated and evade detection, this method has become less effective. To address this issue, new methods according to machine learning (ML) [11]–[14] have been developed to detect malware. These methods typically use deep learning and supervised learning algorithms, such as convolution neural networks (CNNs) and support vector machines (SVM), to analyze the behavior and structure of software and identify patterns that indicate malware.

Convolution neural networks [15]–[17] are used often in processing of still images and moving video but can also be applied to analyzing code. They can identify patterns in the data by analyzing multiple layers of features, which allows them to detect malware even when it is disguised or encrypted [18]. SVM are a kind of the learning technique known as supervised learning [19]–[22], and they may be used to classify data into a variety of categories. They achieve this via creating a boundary, or "hyperplane", that separates the data into different classes. In the case of malware detection, an SVM is capable of being taught to identify the characteristics of malware and then used to label newly developed software as either malicious software or non-malware.

The state of art models [23]–[25] in malware prediction are signature-based recognition. It is a traditional technique for identifying malicious software, which entails making a comparison between code of a program against a database of recognized signatures of malicious software. If a match is found, the program is flagged as malware. Having said that, this approach is becoming less comparable to malicious software has evolved to avoid being discovered by changing its pattern of behavior. Behavior-based detection: This method analyses the behavior of software to identify patterns that indicate malware. It can detect malware that is designed to evade signature-based detection methods by analyzing the actions it takes on a computer.

Heuristics-based detection method identifies potentially malicious software by following a predetermined set of guidelines, sometimes known as heuristics. It is possible to utilize it to detect unknown or new malware variants. Machine learning-based detection method uses algorithms such as deep learning, supervised learning, and unsupervised learning, to analyze the behavior and structure of software and identify patterns that indicate malware [26]. This method can be more effective than traditional methods, but it requires large amounts of data to train the models, which can be a challenge to collect and process. Hybrid-based detection: This approach brings together a number of techniques to make things better overall performance and accuracy of malware detection. An example of this is combining deep learning with signature-based detection; this approach makes advantage of deep learning to pull out characteristics from the code of a program, and a signature-based approach to compare the features with a database of known malware signatures. While these methods can be effective in detecting malware, they do have some limitations. They require large amounts of data to train the models, which can be a challenge to collect and process. Additionally, they can take a long time to train, which can be a bottleneck in the deployment of malware detection systems.

The current hybrid convolution neural network-long short-term memory network (CNN-LSTM) model is a machine learning algorithm [27], it combines the strengths of LSTM and CNNs neural networks in order to speculate on the possibility of a personal computer being infected with malware. This model is designed to analyze various system factors pertaining to location, firmware version, operating system, and antivirus software to determine the probability of malware detection. The model uses CNNs to extract important characteristics derived from the raw data, which are then fed into LSTM networks to make a prediction. CNNs are a form of neural network that is for image and video processing tasks because they can locate recurring elements and distinguishable characteristics in photographs by analyzing their spatial relationships. When it comes to the detection of malware, CNN is capable of being taught to recognize patterns in the data. system variables that are indicative of malware infection.

LSTM networks, on the other hand, are ideal for processing data in a consecutive order because they can remember information from preceding time steps and include it into your projections about subsequent periods of time. In the example of the detection of malware, the LSTM can use the output from the CNN to form an estimate or forecast about the likelihood of malware infection based on the current system variables and their historical patterns. The system variables that are used in this model have been identified as the most important variables for predicting malware detection, and they are given higher weights in the model. These variables include the location of the computer, the firmware version, the operating system, and the anti-virus software. The location of the computer is important because certain locations may be more prone to malware infections than others. The firmware version and operating system are also critical because they can have vulnerabilities that can be exploited by malware. Finally, the anti-virus software is crucial because it can help to detect and remove malware from the system.

The proposed hybrid CNN-LSTM model is a powerful tool for predicting malware detection based on system factors pertaining to location, firmware version, operating system, and antivirus software. By combining the strengths of CNNs and LSTMs, this model can identify patterns and historical trends in the data to make accurate predictions about the likelihood of malware infection. The rest of the paper is organized as follows: section 2 illustrates existing literature, section 3 describes proposed model, section 4 illustrates results and discussions, and section 5 concludes the paper.

An approach for automatically detecting malware it employs CNN as its data processing mechanism and other machine learning methods has been proposed by Yeo *et al.* [1]. The suggested technique is more reliable than the approaches that are already in use since it makes use of 35 distinct characteristics that are retrieved from packet flow rather than just port numbers or protocols. The performance of this model was

evaluated with the use of data from the Stratosphere IPS project, and the results indicated that it had an accuracy of more than 85%, as well as precision and recall for all classes, when CNNs and random forests were employed for classification.

A approach for the identification of malware that is achieved by extensive study and combines static and dynamic analysis of the samples has been proposed by Huang *et al.* [2]. The approach that was proposed makes use of Cuckoo Sandbox for the purpose of doing dynamic analysis on the sample files, converts the results of that investigation into an image in accordance with an algorithm that was designed, and then trains a convolutional neural network (VGG16) with both static images and hybrid visualization images. In the end, its performance is examined by determining how well it can identify previously undiscovered forms of malware.

Unsupervised learning is used in the malware detection approach that is presented in Jin *et al.* [3]. This strategy makes use of deep learning to transform harmful files into pictures, which are then passed through autoencoders to determine if the program in question is benign or malicious. When it comes to the detection of malware, the suggested model obtains an accuracy of 93% with considerably superior F1-score values. This is much greater than the accuracy achieved by standard approaches, which need a significant quantity of training data and are prone to labelling problems.

The work done by Patil and Deng [4] gives a framework for the quicker and more accurate discovery of malicious software by applying machine learning techniques. In order to extract features from malware files, it leverages system calls, operational codes, section codes, and byte codes are all examples of feature-sets. These characteristics are then fed into a variety of machine learning techniques (such as shallow ML models) or deep neural networks (DNNs). The results of the experiments demonstrated that the maximum accuracy could be achieved by using system call feature vectors, with DNN performing much better than conventional shallow ML techniques.

The research presented by Kotian and Sonkusare [5] article centers on the use of deep learning models for the purpose of detecting malware in a cloud-based setting. It has come to everyone's attention that concurrent with the rise in demand for cloud services, there has also been a rise in the number of malicious assaults. The dataset that was used for this investigation was divided into central processing unit (CPU), memory, and network parameters, which were at that time used as input for a variety of deep learning models, including CNN and the LeNet-5 CNN model. Standardization, in conjunction with hyperparameter tuning carried out using GridSearchCV or HyperParameter tuning, was crucial in achieving a high level of detection accuracy (more than 95% overall).

Reinforcement learning is proposed as an innovative method for Rathore *et al.* [6] revolutionary approach to the building of strong Android models for detecting malicious software that are resistant to hostile competition assaults. We have suggested two distinct varieties of evasion attack policies, namely one policy-based evasion attack to be used for the case of perfect knowledge and a multi-policy evasion attack is being prepared for the scenario of limited knowledge. The objective is to discover the flaws that are present in the malware detection models that are already in use and then to devise countermeasures, such as model retraining, defensive distillation, or generative adversarial networks (GANs), to protect against such flaws.

A technique for detecting malicious processes in terminals that may be infected with malware was suggested by Tobiyama *et al.* [7]. This approach is based on the behavior of processes in terminals that may be infected. While RNN was taught to extract features from behavioral logs, CNN was used to categorize feature pictures created by these retrieved features. Feature images were generated by training the RNN to extract features from behavioral logs. The findings of our study revealed that when we used an image size of 30×30 pixels, we were able to acquire an AUC score of 0.96. This is a very high performance in comparison to other approaches that are currently available.

ElMouatez MalDozer is an automated system for the detection of Android malware and the attribution of families of malware that was presented by Karbab *et al.* [8]. The technology depends on sequence categorization using deep learning methods. MalDozer automatically discovers dangerous patterns and extracts them beginning with the unprocessed order of API method calls made by the program. This allows it to identify malware on Android devices. Not only can it be installed on servers, but it can also be used in mobile devices or even in internet of things (IoT) devices, all with the same high level of accuracy: Score on the F1-test between 96% and 99%, false positive rate between 0.06% and 2%.

A framework for deep learning that is both scalable and hybrid, designed for real-time deployments is proposed by Vinayakumar *et al.* [9] as a technique with regard to the efficient visual detection of malware. The suggested technique detects, classifies, and categorizes malware using public and private datasets by using traditional machine learning algorithms (MLAs) in addition to deep learning architectures. The accuracy of the results acquired from these algorithms is improved by using a unique image processing approach in conjunction with MLAs and deep learning models.

Deep transfer learning is used by AlGarni *et al.* [10] in their innovative classification model for malware referred to as malware classification with fine-tune CNN (MCFT-CNN). This model classifies malware pictures into the families to which they belong. The architecture of ResNet50 is improved by the model with the inclusion of an extra dense layer that is completely linked, and the model then supplies the output of the dense layer, together with the information from ImageNet to a SoftMax layer that classifies harmful data. Experiments have shown that this method delivers excellent accuracy and short prediction time on two benchmark datasets, which indicates consistent performance and generalizability towards comparable data sets including unknown harmful samples even if they were generated using sophisticated evasive tactics.

The research conducted by Andrade *et al.* [11] presents a sizable new dataset for the categorization of malware, which has been made accessible to the public. Next, the authors provide a model that has the potential to teach a multiclass classification RNN to use recurrent neural networks, more precisely a LSTM, using the dataset. When tested on previously unknown programs, this LSTM achieves an accuracy of 67.60%, encompassing six classes and five distinct forms of malicious software. In addition, other models such as gated recurrent unit (GRU) and convolutional neural networks are also discussed in this article, but they focus mainly on binary classifications rather than multi-class ones like our proposed approach does here.

2. METHOD

This section describes the proposed hybrid CNN-LSTM model that combines CNNs and LSTMs to speculate on the possibility of a personal computer being infected with malware. Figure 1 illustrates CNN-LSTM architecture for Malware prediction. It analyses system variables such as location, the firmware version currently installed, the operating system, and the anti-virus software to determine the probability of malware detection. CNN extracts feature from the input data, while LSTM processes sequential data and uses the output from the CNN to make predictions about the likelihood of malware infection based on current and historical patterns. The system variables with the highest weight in the model are the location of the computer, version of the anti-virus software, the operating system, and the firmware. These variables are critical in predicting malware detection, as certain locations may be more prone to infections, firmware and operating systems can have vulnerabilities, and anti-virus software can detect and remove malware from the system. Overall, the hybrid CNN-LSTM model is a powerful tool for accurately predicting malware detection based on system variables.

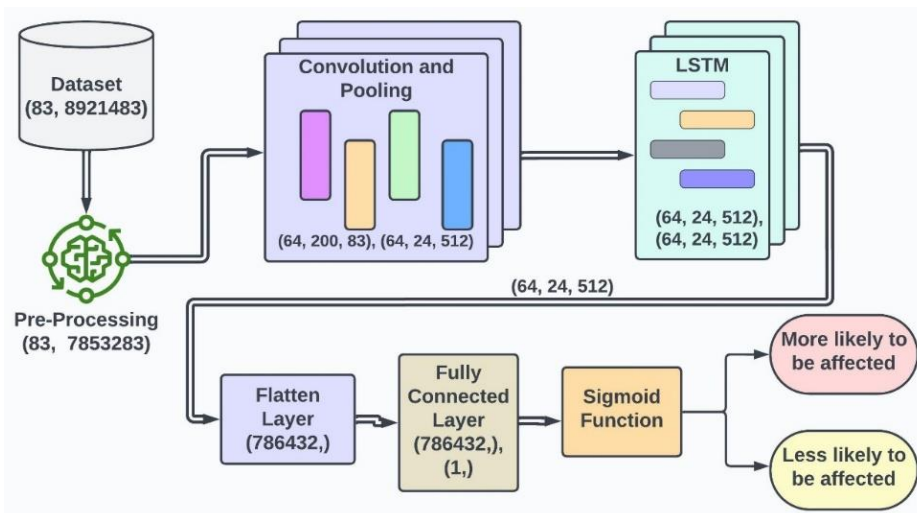


Figure 1. CNN-LSTM architecture for malware prediction

The convolutional layers are followed by pooling layers, which reduce the dimensionality of the data. This is done by taking the maximum or average value of a group of adjacent pixels, for example. This helps to reduce the computational cost of the model and makes the features more robust to small changes in the input data. The output of the CNN component is a set of features that have been extracted from the input data. These features are designed to be relevant for the task of predicting malware. The extracted features are then fed into the LSTM component of the model. LSTM networks are designed to handle sequential data, making them well-suited for time-series data, such as the sequence of system features in a malware sample.

The LSTM component processes the sequence of features over time to make predictions about the malware. The LSTM network has a series of memory cells that can store information over a long period of time, and gates that control when to read or write to these cells. This allows the LSTM to keep track of the sequence of features and make predictions based on the current feature and the history of previous features.

The final output of the model is a prediction of whether the input sample is malware or not. This output is produced by passing the output of the LSTM component via a layer that is completely linked and a sigmoid function. The output of the sigmoid function will be a probability between 0 and 1, indicating the likelihood that the input sample is malware. By combining the strengths of CNNs and LSTMs, this model can create predictions based on the input data and extract useful characteristics from those data based on those features, even when the input data is sequential in nature. In CNN, the mathematical equations mainly consist of convolution, activation, and pooling operations, for instance for a convolution operation:

$$\text{Convolution}(\text{input}, \text{kernel}) = \text{input} * \text{kernel} + \text{bias}$$

Activation functions like ReLU, Sigmoid, and tanh are usually applied after the convolution operation. In LSTM, the mathematical equations include the input gate $i(t)$, forget gate $f(t)$, output gate $o(t)$, candidate gate $g(t)$, memory cell $c(t)$, and the output $h(t)$:

$$i(t) = \text{sigmoid}(W_i \times x(t) + W_i \times h(t-1) + b_i) \quad (1)$$

$$f(t) = \text{sigmoid}(W_f \times x(t) + W_f \times h(t-1) + b_f) \quad (2)$$

$$o(t) = \text{sigmoid}(W_o \times x(t) + W_o \times h(t-1) + b_o) \quad (3)$$

$$g(t) = \text{tanh}(W_g \times x(t) + W_g \times h(t-1) + b_g) \quad (4)$$

$$c(t) = f(t) \times c(t-1) + i(t) \times g(t) \quad (5)$$

$$h(t) = o(t) \times \text{tanh}(c(t)) \quad (6)$$

Algorithm CNN-LSTM

1. Initialize the algorithm: Set up the parameters and variables needed for the algorithm.
2. Initialize the algorithm: Set up the parameters and variables needed for the optimization algorithm.
3. Divide the data into k parts: Split the data into k equal parts, where k is specified by the user.
4. For each round t in the range 1 to z: Perform the k-fold cross-validation.
5. Divide the data into separate sets for training and validation: Use the part of the data specified by round t for validation and the rest of the parts for training.
6. Pre-process the data using techniques such as normalization, data augmentation, and resizing to prepare it for input to the model.
7. Define the CNN-LSTM model architecture:
 - a. Define the convolutional layer:

$$\text{Conv}(i, f) = \text{ReLU}((i * f) + b)$$

where i is the input image, f is the filter/kernel, b is the bias, and $*$ denotes the convolution operation.

- b. Define the LSTM layer using equation (1) to (6)
 - c. Connect the CNN and LSTM layers:
8. Pass the output of the convolutional layer through a Reshape layer to match the input shape expected by the LSTM layer, and then pass the reshaped output into the LSTM layer.
9. Train the CNN-LSTM model: Train the model using the training data and the Adam optimization algorithm.
10. Evaluate the model on the validation set: Evaluate the model on the validation set and store the evaluation result.
11. End the k-fold cross-validation.
12. Choose the best model: Choose the most suitable version in accordance with the evaluation results.
13. Put the model to the test on the test data: Test the best model on the test data using the model Evaluate function.
14. Calculate accuracy: Calculate the accuracy of the model in relation to the test data by comparing the predicted output to the actual output.

3. RESULTS AND DISCUSSION

3.1. Experimental setup

The experimental setup describes the tools and technologies used for conducting the experiments. The experiments are conducted using an 8 GB RAM Windows 10 environment and the Google Colaboratory interface with Python 3. The focus was on evaluating the performance of the proposed CNN-LSTM and existing models for malware analysis.

3.2. Dataset

In this paper the Microsoft malware dataset, which contains a total of 8,921,483 samples of which 44,58,892 are malware instances and 4,462,591 are benign instances with 83 features is used to determine the likelihood that a Windows PC will get infected by a certain family of malicious software by analyzing its individual qualities and characteristics. Combining the heartbeat and threat reports that were gathered by Microsoft Windows Defender, an endpoint security solution, resulted in the generation of the telemetry data that contains the machine infections and the features. This dataset contains rows that each reflect a different kind of machine, and the machine identifier column provides a one-of-a-kind identifier for each machine. The ground truth is shown by the value has Detections, which shows that malware was found on the computer. Using the information and labels included within train.csv, it is your job to make an educated guess as to what the value will be for the column labelled contains detections in the test.csv file for each machine. It uses features like MachineIdentifier, ProductName, and AppVersion, the model uses 70% to train the model and 30% to test the model. Here Table 1 represents the model summary and number of layers used. Type of layers and input and output of each layer are also displayed.

Here, Table 2 describes the comparison of proposed CNN-LSTM and state-of-art models with respect to the performance metrics Accuracy, Precision, Recall, F-Score, and root mean square error (RMSE). The proposed CNN-LSTM model performs better in terms of predicting novel malware with an accuracy of 95%, whereas existing models perform poor (CNN gives 93%, LSTM gives 92%). The proposed CNN-LSTM model performs better in the aspect of handling imbalanced data, it indicates by precision, recall, and F-score whereas existing models fail to handle imbalanced data.

Table 1. Model parameters

Layer	Parameter	Description
Input	Input size	(83, 7853283)
	Input size	(64, 200, 83)
Convolutional layer	Number of filters	128, 256, 512
	Filter size & Stride	3 & 1
	Activation function	ReLU
	Pooling layer	Max Pooling with size 2, stride 2
	Output shape	(64, 24, 512)
LSTM layer	Input shape	(64, 24, 512)
	Number of LSTM units	512
	Activation function	(tanh)
Flatten layer	Output shape	(64, 24, 512)
	Input shape	(64, 24, 512)
Fully connected layer	Output shape	(786432)
	Number of neurons	1
	Activation function	Sigmoid

Table 2. Comparison of proposed and state-of-art models

Model	Accuracy	Precession	Recall	F-score	RMSE
CNN	93	90	90	0.92	0.27
LSTM	92	89	91	0.91	0.22
CNN-LSTM	95	94	94	0.93	0.08

Figure 2 shows the accuracy of the CNN-LSTM model achieves a superior accuracy of 95% at the 99th epoch, outperforming the CNN and LSTM models, that attain accuracies of 93% and 92%, respectively. This heightened accuracy can be attributed to the CNN-LSTM model's ability to effectively capture both spatial and temporal features inherent in malware sequences. While CNNs excel at recognizing spatial patterns, LSTM models specialize in understanding temporal dependencies. The integrated CNN-LSTM architecture harmoniously combines these strengths, offering a more nuanced and comprehensive analysis of malware behavior. The advantages of the CNN-LSTM model become particularly pronounced when handling the Microsoft malware dataset, known for its diverse and dynamic characteristics. The model's capacity to discern complex temporal dynamics and subtle variations in malware patterns contributes to its superior accuracy. CNN and LSTM models, by contrast, exhibit limitations in capturing the intricate dependencies present in temporal sequences, resulting in slightly lower accuracies.

It is noteworthy that the CNN-LSTM model not only surpasses its counterparts in accuracy but also demonstrates enhanced generalization capabilities, proving effective even in scenarios with limited labeled data. This is a crucial advantage, considering the challenges associated with obtaining extensive and diverse labeled datasets for training. While the CNN-LSTM model exhibits notable strengths, it is essential to

acknowledge the inherent limitations of existing models, such as their overspecialization, memorization challenges, and dependence on large, labeled datasets. The CNN-LSTM model addresses these limitations by providing a more holistic and adaptable framework for malware prediction, marking a significant advancement in the field of cybersecurity.

Figure 3 shows the precision values obtained from malware prediction on the Microsoft malware dataset underscoring the distinct advantages of the proposed CNN-LSTM model compared to CNN and LSTM. With precision rates of 90% for CNN, 89% for LSTM, and a notable improvement to 94% for CNN-LSTM, the results reveal the enhanced performance of the novel model. The CNN-LSTM model excels by effectively understanding temporal dependencies within malware sequences, capturing subtle variations over time that contribute to its superior precision. Its ability to combine spatial pattern recognition from CNNs with temporal understanding from LSTM networks results in a synergistic effect, enabling the model to discern complex malware features. On the other hand, existing models face limitations, such as overspecialization in CNNs, primarily designed for image data, and LSTM's challenges with gradient descent during training. These limitations contribute to the lower precision observed in CNN and LSTM models. The comparative analysis with precision values provides a comprehensive view of how the proposed CNN-LSTM model outperforms existing models, making it a promising solution for accurate malware prediction in the Microsoft malware dataset.

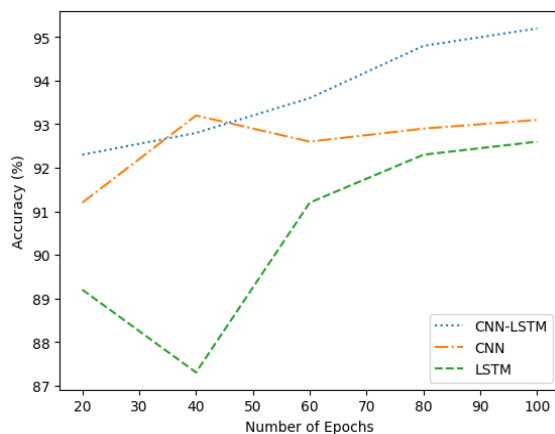


Figure 2. Accuracy

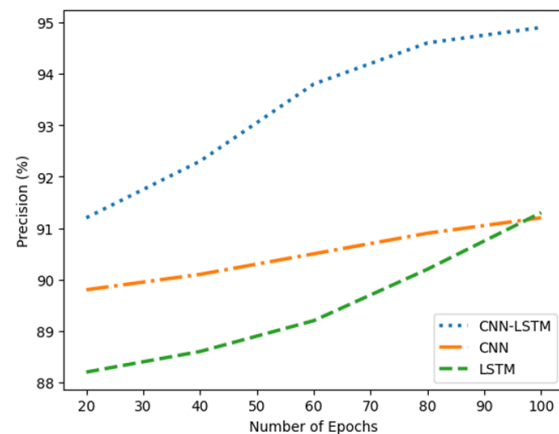


Figure 3. Precision

Figure 4 shows the recall values for malware prediction on the Microsoft malware dataset, the disparities in performance among CNN, LSTM, and the proposed CNN-LSTM model underscore critical differences in their approaches. CNN, with a recall rate of 90%, grapples with the challenge of adaptation in image data, limiting its effectiveness in comprehensively capturing diverse malware patterns. LSTM, achieving a 91% recall rate, faces constraints related to the complexities of gradient descent during training, impacting its ability to discern nuanced temporal dependencies in malware sequences. In contrast, the CNN-LSTM model, boasting a commendable 94% recall rate, the recall is converging from the 25th epoch over the existing and produces 94% at the 100th epoch overcomes these limitations by synergizing the strengths of CNN and LSTM. The model excels in recognizing intricate spatial and temporal correlations within malware data, leading to a substantial improvement in recall. This nuanced analysis illuminates the multifaceted advantages of the proposed CNN-LSTM model, positioning it as a robust solution for enhancing recall in the challenging landscape of Microsoft malware prediction.

Figure 5 examines the F-score values for malware prediction on the Microsoft malware dataset reveals noteworthy distinctions in the performance of CNN, LSTM, and the proposed CNN-LSTM model. CNN, with an F-score of 0.92, demonstrates proficiency but encounters challenges in effectively balancing precision and recall due to its emphasis on image data. LSTM, achieving an F-score of 0.91, grapples with the intricacies of capturing temporal dependencies, leading to a slightly lower overall performance. The proposed CNN-LSTM model, excelling with an F-score of 0.93 converging from the 80th epoch and heights at the 100th epoch, strategically combines the strengths of both CNN and LSTM. This synergy allows for a more harmonized approach to precision and recall, resulting in a superior F-score. The nuanced analysis of F-score values positions the CNN-LSTM model as a well-balanced solution, effectively navigating the complexities of Microsoft malware prediction with improved precision and recall.

Figure 6 shows the RMSE values for Microsoft malware dataset prediction emphasizing the superiority of the proposed CNN-LSTM model over CNN and LSTM. CNN, with an RMSE of 0.27, indicates a moderate level of prediction error, attributed to its emphasis on specific features, which may not fully capture the nuanced patterns in malware data. LSTM, achieving an RMSE of 0.22, shows improved predictive accuracy but struggles with handling features related to temporal dependencies, limiting its capacity to discern evolving trends in malware behavior. In contrast, the CNN-LSTM model excels with an impressively low RMSE of 0.08 at the 100th epoch, showcasing its capability to address both spatial and temporal intricacies. This emphasizes the CNN-LSTM model's effectiveness in providing accurate predictions for Microsoft malware instances, highlighting its potential to overcome the limitations of existing models and enhance overall security measures.

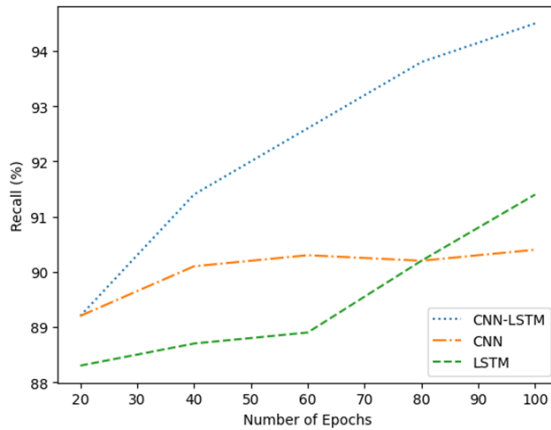


Figure 4. Recall

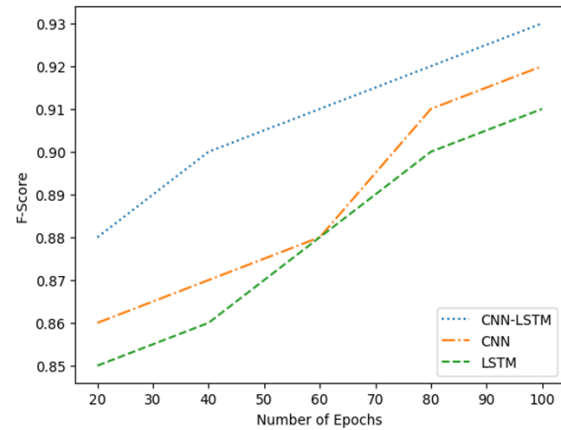


Figure 5. F-score

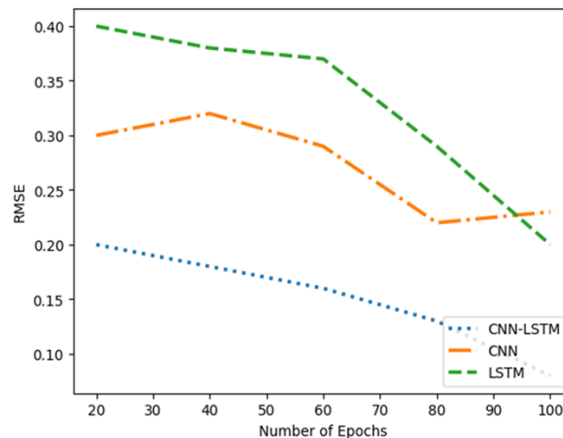


Figure 6. Root mean square error

4. CONCLUSION

The identification of malicious software is an essential step in assuring the users' safety, network infrastructure, and data privacy. However, detecting malicious activity remains a challenging task due to the emergence of new forms of malware and cybercrimes, as well as the vast amount of data with shifting patterns. The widely used bag-of-words (BOW) method fails to capture the spatial and sequence information that is essential for detecting malware patterns. To address this challenge, researchers have developed cutting-edge algorithms for detecting malicious activity, including the hybrid CNN-LSTM combination models proposed in this study. The models leverage the temporal modelling of LSTM and the potentiality of CNN to derive geographical data-derived information in order to forecast whether or not a personal computer will get infected with malicious software or not based on system variables such as location, the firmware version currently installed, the operating system, and the anti-virus software. The proposed models were

trained and evaluated using Microsoft's malware dataset, and their performance when comparing with other approaches. The findings from the experiments indicate that the superiority of the suggested models over traditional MLP, CNN, and LSTM models, with the LSTM-CNN model achieving existing performance with an accuracy of 95% on the Microsoft malware dataset. Notably, the study found that the most important variables for predicting malware detection were related to the system variables of location, the firmware version currently installed, the operating system, and the anti-virus software, which had the highest weight in the model's predictions. This finding suggests the importance of considering system variables when developing malware detection models. Overall, the proposed hybrid CNN-LSTM models outperformed existing models in relation to precision, recall, F-score, and loss.




REFERENCES

- [1] M. Yeo *et al.*, "Flow-based malware detection using convolutional neural network," in *2018 International Conference on Information Networking (ICOIN)*, Jan. 2018, pp. 910–913, doi: 10.1109/ICOIN.2018.8343255.
- [2] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *Journal of Signal Processing Systems*, vol. 93, no. 2–3, pp. 265–273, Mar. 2021, doi: 10.1007/s11265-020-01588-1.
- [3] X. Jin, X. Xing, H. Elahi, G. Wang, and H. Jiang, "A malware detection approach using malware images and autoencoders," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Dec. 2020, pp. 1–6, doi: 10.1109/MASS50613.2020.00009.
- [4] R. Patil and W. Deng, "Malware analysis using machine learning and deep learning techniques," in *2020 SoutheastCon*, Mar. 2020, pp. 1–7, doi: 10.1109/SoutheastCon44009.2020.9368268.
- [5] P. Kotian and R. Sonkusare, "Detection of malware in cloud environment using deep neural network," in *2021 6th International Conference for Convergence in Technology (I2CT)*, Apr. 2021, pp. 1–5, doi: 10.1109/I2CT51068.2021.9417901.
- [6] H. Rathore and S. K. Sahay, "Towards robust android malware detection models using adversarial learning," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Mar. 2021, pp. 424–425, doi: 10.1109/PerComWorkshops51409.2021.9430980.
- [7] S. Tobiya, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Jun. 2016, pp. 577–582, doi: 10.1109/COMPSAC.2016.151.
- [8] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: a framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, pp. S48–S59, Mar. 2018, doi: 10.1016/j.diin.2018.01.007.
- [9] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- [10] M. D. AlGarni, R. AlRoobaea, J. Almotiri, S. S. Ullah, S. Hussain, and F. Umar, "An efficient convolutional neural network with transfer learning for malware classification," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–8, Oct. 2022, doi: 10.1155/2022/4841741.
- [11] E. de O. Andrade, J. Viterbo, C. N. Vasconcelos, J. Guérin, and F. C. Bernardini, "A model based on LSTM neural networks to identify five different types of malware," *Procedia Computer Science*, vol. 159, pp. 182–191, 2019, doi: 10.1016/j.procs.2019.09.173.
- [12] B. Vignau, R. Khoury, S. Hallé, and A. Hamou-Lhadj, "The evolution of IoT Malwares, from 2008 to 2019: survey, taxonomy, process simulator and perspectives," *Journal of Systems Architecture*, vol. 116, Jun. 2021, doi: 10.1016/j.sysarc.2021.102143.
- [13] M. Baig, P. Zavarisky, R. Ruhl, and D. Lindskog, "The study of evasion of packed pe from static detection," *World Congress on Internet Security (WorldCIS-2012)*, pp. 99–104, 2012.
- [14] D. W. Fernando, N. Komninos, and T. Chen, "A study on the evolution of ransomware detection using machine learning and deep learning techniques," *IoT*, vol. 1, no. 2, pp. 551–604, Dec. 2020, doi: 10.3390/iot1020030.
- [15] I. Bello *et al.*, "Detecting ransomware attacks using intelligent algorithms: recent development and next direction from deep learning and big data perspectives," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 9, pp. 8699–8717, Sep. 2021, doi: 10.1007/s12652-020-02630-7.
- [16] W. Z. A. Zakaria, M. F. Abdollah, O. Mohd, and A. F. M. Ariffin, "The rise of ransomware," in *Proceedings of the 2017 International Conference on Software and e-Business*, Dec. 2017, pp. 66–70, doi: 10.1145/3178212.3178224.
- [17] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Measurement*, vol. 167, Jan. 2021, doi: 10.1016/j.measurement.2020.108288.
- [18] S. Ben Atitallah, M. Driss, W. Boulila, and H. Ben Ghézala, "Randomly initialized convolutional neural network for the recognition of COVID -19 using X-ray images," *International Journal of Imaging Systems and Technology*, vol. 32, no. 1, pp. 55–73, Jan. 2022, doi: 10.1002/ima.22654.
- [19] S. Ben Atitallah, M. Driss, W. Boulila, A. Koubaa, and H. Ben Ghézala, "Fusion of convolutional neural networks based on Dempster-Shafer theory for automatic pneumonia detection from chest X-ray images," *International Journal of Imaging Systems and Technology*, vol. 32, no. 2, pp. 658–672, Mar. 2022, doi: 10.1002/ima.22653.
- [20] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proceedings of the International Conference on Artificial Neural Networks*, 2018, pp. 270–279.
- [21] G. Vrbancic and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196197–196211, 2020, doi: 10.1109/ACCESS.2020.3034343.
- [22] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: a survey," *Information Fusion*, vol. 37, pp. 132–156, Sep. 2017, doi: 10.1016/j.inffus.2017.02.004.
- [23] O. Sagi and L. Rokach, "Ensemble learning: a survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, Jul. 2018, doi: 10.1002/widm.1249.
- [24] M. Nisa *et al.*, "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Applied Sciences*, vol. 10, no. 14, Jul. 2020, doi: 10.3390/app10144966.
- [25] J. Hemalatha, S. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient DenseNet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, Mar. 2021, doi: 10.3390/e23030344.




- [26] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Software Quality Journal*, vol. 26, no. 3, pp. 891–919, Sep. 2018, doi: 10.1007/s11219-017-9368-4.
- [27] A. P. Gopi, M. Gowthami, T. Srujana, S. G. Padmini, and M. D. Malleswari, "Classification of denial-of-service attacks in IoT networks using AlexNet," in *Human-Centric Smart Computing: Proceedings of ICHCSC 2022*, 2023, pp. 349–357.

BIOGRAPHIES OF AUTHORS






Nagababu Pachhala    is currently a Ph.D. research scholar at Annamalai University, Chidambaram, Tamil Nadu, India. He received his M.Tech. degree in information technology from Nalanda Institute of Engineering and Technology, Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India in 2012, and B. Tech degree in computer science and information technology from Nalanda Institute of Engineering and Technology, Jawaharlal Nehru Technological University, Hyderabad, Andhra Pradesh, India in 2006. His research interests include malware analysis, malware detection, machine learning, and cloud computing. He can be contacted at email: nagababupachhala2024@gmail.com.



Subbaiyan Jothilakshmi    received the B.E. degree in electronics and communication engineering from Govt. College of Engineering, Salem in 1994. She received the M.E. degree in computer science and engineering from Annamalai University in the year 2005. She has been with Annamalai University, since 1999. She completed her Ph.D. degree in computer science and engineering at Annamalai University in 2011. She published 52 papers in international journals and conferences. Her research interests include speech processing, image and video processing, pattern classification and machine learning. She can be contacted at email: jothi.sekar@gmail.com.



Bhanu Prakash Battula    completed PDF (Post Doctoral Fellowship) in 2021 from Malaysia. He completed his Ph.D. from Acharya Nagarjuna University in 2014 and Post graduation from Acharya Nagarjuna University in 2008. He published 48 publications in reputed National and International Journals. He has 6 patents in machine learning and published 6 books in machine learning and computer vision. He got two National Awards "Bharat Excellence" and "Mother Teresa Teaching Excellence" by Friendship Forum of India, New Delhi by the Governor of Sikkim in 2017. He can be contacted at email: prakashbattula33@gmail.com.