A Deep Learning Approach for Detecting Malware Using Autoencoder

Venu Madhav Panchagnula, Ch. Satya Keerthi N.V.L, S. Surekha, R. Sujatha, Duggineni Veeraiah, Eluri Ramesh, Lakshmi B.

***Abstract*—Research in the field of malware detection is currently leaning toward methods based on artificial intelligence algorithms due to the increasing limitations of traditional detection methods and the increasing accuracy of these methods. This study therefore introduces a novel approach to malware detection. This method integrates a deep learning model's autoencoder network with a grayscale graphic depiction of malware. By looking at the autoencoder's reconstruction error, we may assess if the grey-scale picture approach is feasible to use for malware detection. Additionally, it differentiates between malicious and safe software by utilizing the autoencoder's dimensionality reduction capabilities. Using the Android dataset, our newly introduced detection model attained a 95% accuracy and maintained a consistently high F-score of approximately 95%. This outperformed more conventional methods of machine learning detection.**

Index Terms—Malware detection, autoencoders, malware images, machine learning.

I. Introduction

IN an era dominated by digital interconnectedness, the ubiquity of computing devices has propelled society into an age of unprecedented technological advancement. However, this progress has also ushered in a parallel evolution in cyber threats, particularly in the form of malware. Malicious software, encompassing a wide array of code designed to infiltrate, damage, or exploit computer systems, poses a persistent challenge to information security. Traditional methods of malware detection, reliant on static signatures, have struggled to keep pace with the rapid mutation and sophistication exhibited by contemporary malware variants.

The dynamic nature of modern malware, characterized by polymorphic and metamorphic traits, demands a paradigm shift in detection methodologies. As a response to this challenge, deep learning, a subset of machine learning, has emerged as a promising avenue for developing adaptive and robust malware detection systems. Within the realm of deep learning, autoencoders, a class of neural networks primarily employed for unsupervised learning tasks, offer a unique approach to learning data representations and detecting anomalies [1] – [3].

The primary objective of this research is to investigate the efficacy of autoencoders in the context of malware detection. Autoencoders, known for their ability to capture latent features within data, present an attractive option for discerning the subtle patterns indicative of malicious activity. By exploring the potential of autoencoders in this domain, we seek to contribute to the development of more resilient and responsive defenses against the ever-evolving landscape of malware threats [4] – [5]. Through rigorous experimentation and analysis, this research aims to: Evaluate the performance of autoencoders in detecting diverse malware samples. Compare the results with traditional signature-based methods and other deep learning approaches. Assess the adaptability and scalability of the proposed autoencoder-based malware detection system. Identify the strengths and limitations of autoencoders in addressing the challenges posed by contemporary malware [6] – [7].

A. Traditional Malware Detection

Historically, malware detection has heavily relied on signature-based methods, where predefined patterns or signatures of known malware are compared against files or system activities. While effective against known threats, this approach falters in the face of polymorphic and metamorphic malware. Polymorphic malware dynamically alters its code while retaining its original functionality, challenging static signature-based systems. Metamorphic malware goes further by completely changing its code structure, rendering signature-based detection even less effective [8] – [9]. The limitations of traditional methods necessitate the exploration of more adaptive and intelligent approaches to malware detection [10] – [11].

Manuscript received January 6, 2024; revised July 4, 2024.

Venu Madhav Panchagnula is Assistant Professor of Electronics and Communication Engineering Department, Prasad V Potluri Siddhartha Institute of Technology, Vijayawada, Andhra Pradesh, India. (e-mail: venumadhav@pvpsiddhartha.ac.in)

Ch. Satya Keerthi N.V.L is Assistant Professor of Computer Science and Engineering Department, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India. (e-mail: satyakeerthinvl@gmail.com)

S. Surekha is Assistant Professor of AIML Department, Vignana Bharathi Institute of Technology, Ghatkesar, Hyderabad, India. (e-mail: surekhasatram@gmail.com)

R. Sujatha is Assistant Professor of Information Technology Department, M. Kumarasamy College of Engineering, Karur, India. (e-mail: sujathar.it@mkce.ac.in)

Duggineni Veeraiah is Professor of Computer Science and Engineering Department, Lakireddy Bali Reddy College of Engineering, Mylavaram, Andhra Pradesh, India. (e-mail: veeraiahdvc@gmail.com)

Eluri Ramesh is Assistant Professor of Computer Science and Engineering Department, R.V.R & J. C. College of Engineering, Guntur, India. (e-mail: eluriramesh@rvrjc.ac.in)

B. Lakshmi is a Scholar of Electronics and Communication Engineering Department, Rajiv Gandhi University of Knowledge Technologies, Nuzvid, Andhra Pradesh, India. (e-mail: blakshmith@gmail.com)

B. Deep Learning in Malware Detection

The limitations of conventional approaches can be somewhat addressed by deep learning techniques, which can automatically learn hierarchical representations from data. Improved malware detection systems have been made possible through the use of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to extract useful information from malware samples. While RNNs are great at modeling sequential patterns in code behavior, CNNs are great at capturing spatial relationships in data, which makes them ideal for evaluating malware representations in binary or picture form [12] - [13]. Malware, however, is constantly developing, necessitating new approaches to detection.

C. Autoencoders in Anomaly Detection

Autoencoders, a type of unsupervised deep learning model, have gained attention for their efficacy in anomaly detection across various domains. An autoencoder comprises an encoder and a decoder, with the latent space between them serving as a compressed representation of input data. During training, the autoencoder learns to reconstruct the input data accurately, making it adept at capturing intrinsic features while highlighting anomalies as deviations from the learned norm [14] – [15]. Previous studies have demonstrated the utility of autoencoders in identifying anomalous patterns in diverse datasets, from medical images to network traffic. This adaptability makes autoencoders a compelling candidate for detecting the subtle and dynamic characteristics of malware [16] – [17]. The literature suggests that autoencoders, with their capacity to discern anomalies in an unsupervised manner, hold significant promise for addressing the challenges posed by polymorphic and metamorphic malware, providing a foundation for the exploration of their application in malware detection [18] – [19].

II. Methodology

A. Data Collection

To evaluate the effectiveness of autoencoders in malware detection, a diverse and representative dataset of malware samples is essential. The dataset should encompass a variety of malware types, including polymorphic and metamorphic variants, to ensure the robustness and generalizability of the trained model. Publicly available malware repositories, such as the Malware Genome Project or the Microsoft Malware Classification Challenge dataset, can serve as valuable sources for this purpose [20] – [22].

B. Auto encoder Architecture

The design of the autoencoder architecture is a critical component of the methodology. The autoencoder should be capable of learning compact and meaningful representations of the input data, distinguishing between normal and malicious patterns effectively. The architecture may include multiple layers with varying activation functions, and hyper parameters such as the learning rate and batch size must be carefully tuned through experimentation to optimize the model's performance [23]. Given the dynamic nature of malware, a robust architecture may incorporate convolutional layers for spatial feature extraction, recurrent layers for capturing sequential dependencies, or a combination of both to enhance the model's ability to detect diverse types of malware [24].

C. Training and Evaluation

So that the model may be trained on a well-rounded sample of malware and then tested on new cases, the dataset is split into two parts: the training set and the testing set. By reducing reconstruction errors, the autoencoder learns to encode data with regular patterns during training. A number of metrics are used to assess the model's performance, including recall, accuracy, precision, and F1-score [25]. In order to evaluate the autoencoder's anomaly detection capabilities, we compare it to other deep learning algorithms, such as RNNs or CNNs, and to more conventional signature-based methods. The model's resilience to changes in the dataset, including unequal distribution of classes or the appearance of new viruses, is also investigated [26]. To prevent overfitting and make sure the results are robust, you can use cross-validation methods like k-fold cross-validation.

III. APPROACH

Our malware detection approach is built on top of the automatic encoder network. Our malware detection method's general structure and primary duties are shown in Figure 1. The process begins with decompiling the APK files, which turns benign and malicious files into appropriate greyscale graphics. The binary codes are extracted from software methods and converted to decimal data bytes before being filled with pixel values. After that, in order to finish two tasks, the grayscale photos are fed into two deep learning networks. In the first, known as automatic encoder network - 1(AE-1), we analyze the viability of representing software features using greyscale images; in the second, known as automatic encoder network - 2(AE-2), we classify malicious software from benign software.

A. The Autoencoder's Structure

One kind of unsupervised neural network used in deep learning models is the autoencoder network structure. Figure 2 shows that it consists of two networks: one for encoding and one for decoding. The encoding network compresses the data and decreases the number of dimensions, while the decoding network successfully recreates the input. The goal of training and updating the parameters of an autoencoder network is to minimize the loss function, which is defined as the difference between the input and the corresponding output of the model.

We created two model constructions, AE-1 and AE-2, with the former being designed initially and the latter following. The primary goal in creating the AE-1 network was to evaluate potential feature extraction techniques for greyscale photos; the primary goal in creating the AE-2 network was to identify malware. We recommended the AE-

2 network as an upgrade from the AE-1 network because the former had more problems and was less stable during the experimental portions of the classification task. We built the two networks for this very reason. You should be aware that, unlike the unsupervised AE-1 network, which does not use labels, the AE-2 network requires them during training. Malicious and benign software samples are tagged accordingly.

Model AE-1's architecture includes convolutional, pooling, and up-sampling layers, as illustrated in Figure 3.

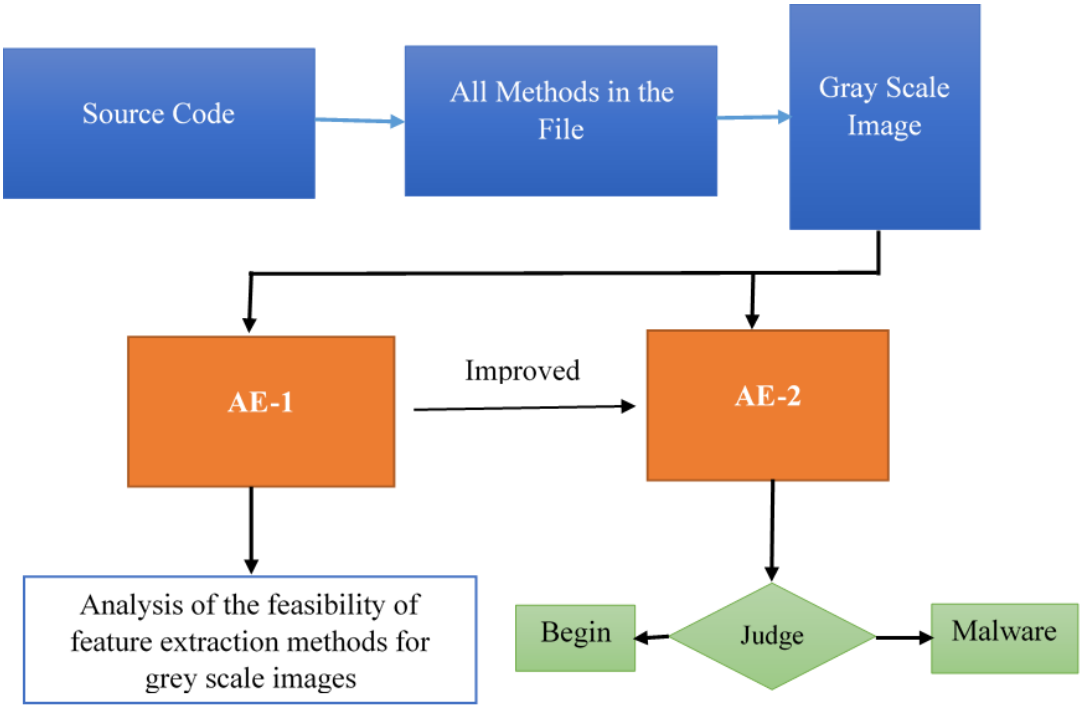

Figure 1. A summary of our suggested method

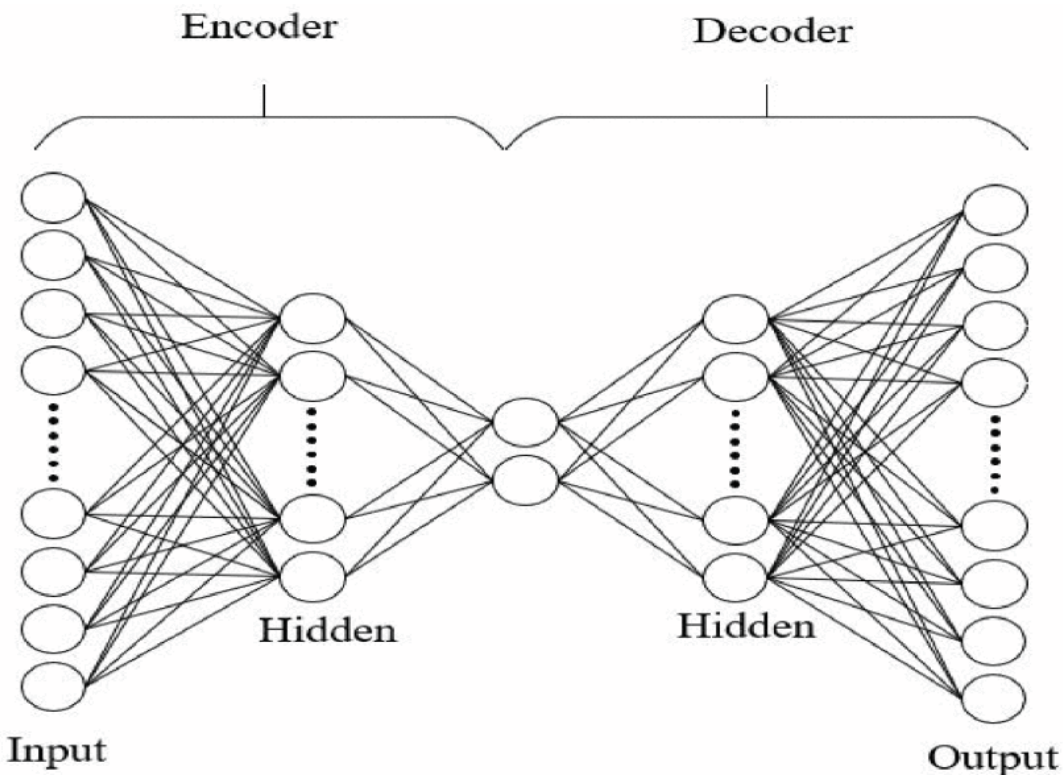

Figure 2. An autoencoder's schematic depiction

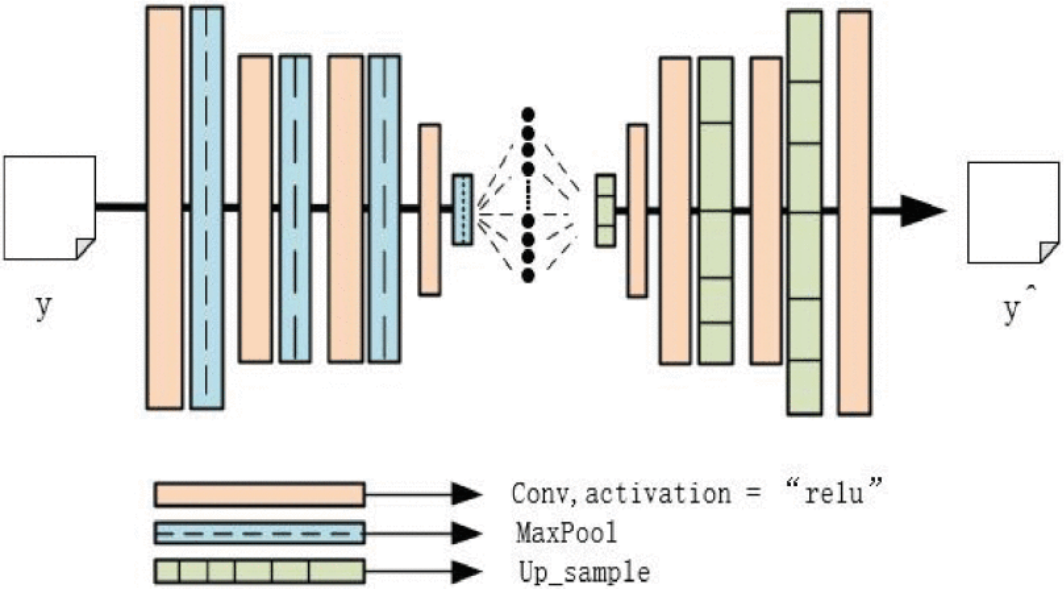

Figure 3. (AE-1) The first automated encoder construction

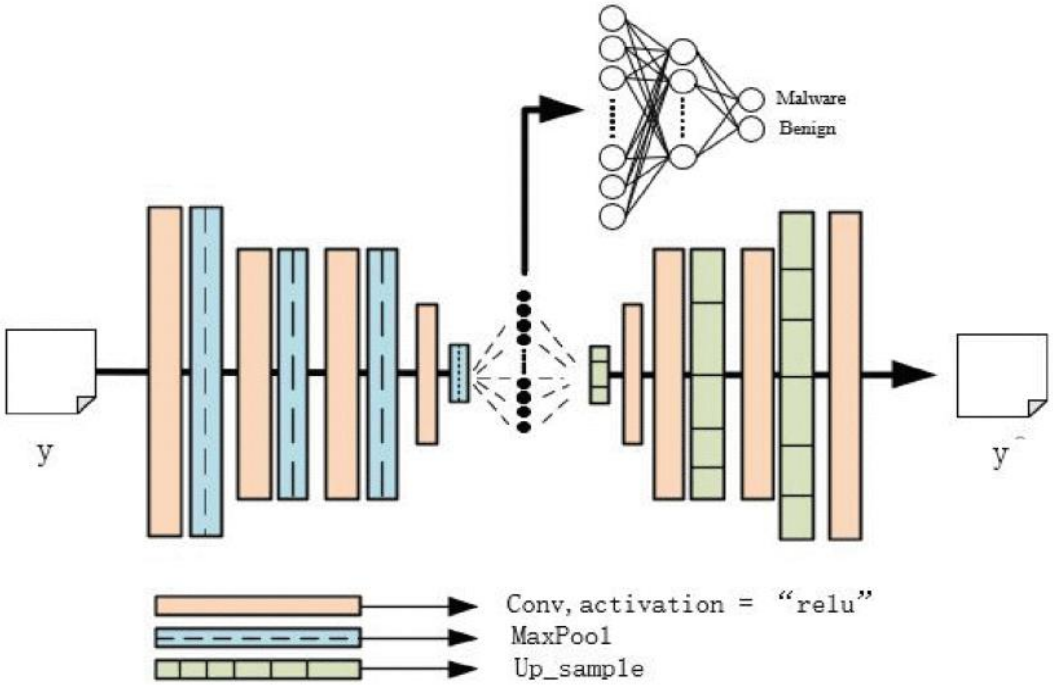

Figure 4. The AE-2 structure is the second automated encoder structure

Figure 4 depicts the structure of model AE-2, which has an autoencoder network topology identical to model AE-1. Our primary distinction is the incorporation of an external multi-layer perceptron network into our classification and experimental processes. We begin by pre-training model AE-1 to extract high-dimensional features that are relevant to malicious and benign software. Then, we train the multi-layer perceptron network using the output from model AE-1's hidden layer. The multiple-layer perceptron network generates two-dimensional vectors to perform the task of classifying malware and benign software.

IV. Results and Discussion

The results of the experiments conducted to evaluate the efficacy of the autoencoder-based malware detection approach are presented and discussed in this section. The present section focuses on information relevant to the experimental setup and is organized into three sections for the following reason: the setting of the experiment setup, the collection of data, and the training details. Based on their intended use, we classified the datasets as follows: (1) Dataset-1, which contains 8,121 malicious programs and 2,000 benign ones; it is used to train and evaluate AE-1 models. (2) Dataset-2, which includes 8121 malicious programs and 7015 safe ones, is utilized for AE-2 model training, validation, and testing. (3) Dataset-3, which

contains 5,384 malicious programs and 5,000 safe ones, is utilized to examine how well the AE-2 model detects unknown software. It should be mentioned that when we split Dataset-2 and Dataset-3, we purposefully included older software samples in Dataset-2 for training purposes, such as malware from 2016, and we included newer releases in Dataset-3, such as 2017, 2018. This will make it easier to analyze the model's performance in the future when it identifies newly released softwear samples by simulating that situation. To analyze the autoencoder's performance in reconstructing feature pictures, the AE-1 network is employed; Table 2 displays the specific characteristics of the AE-1 model. During training, we employ the Adam optimization method with an epoch of 100 and a learning rate of 1e-4.

The AE-1 network undergoes training using the DTrain dataset and subsequently undergoes testing using the DTest_mal and DTest_benign datasets, which contain malicious and benign software, respectively. In order for a test set to have a minimal reconstruction error, the new input must be similar to the input of the training dataset. Conversely, if the new inputs deviate from the inputs used in training the dataset, a noticeable reconstruction error will be observed in this test set. The main objective of our experiment is to investigate the significant difference in error data produced by these two test sets after AE-1. The presence of extensive redundancy in the software dataset and the distinct functional characteristics exhibited by various malware families in the malware dataset can result in significant fluctuations in experimental outcomes. This is because our hypothesis is founded on the notion that malware is uniformly similar, while benign software is not. Consequently, we place less importance on the exact errors exhibited by the two test sets and focus more on the comparative disparities between them.

The responsibility of evaluating the performance of the detection model lies with the AE-2 network. We partitioned Dataset-2 into two equal parts, allocating 80% for training and 20% for testing. During the training process, we employed k-fold cross-validation with a value of k equal to 6 in order to train and validate the training set. Consequently, we allocated 5/6 of the training set for training purposes and reserved 1/6 for validation. We conducted this procedure on six occasions prior to calculating the average. The test set is used for testing purposes during the entire testing procedure. The duration of training is quantified in minutes. AE-2's training utilized the Adam optimization technique with a learning rate of 0.0001 and 100 epochs.

We evaluate the effectiveness of our strategy by comparing the overall error distribution in harmful and benign reconstructions of malware images. Figure 5 displays the error distributions of the combined test sets. The Y-axis indicates the normalized reconstructed error value generated by each program following the encoder network. To normalize the image, the sum of the error values for all pixels in the malware feature picture is computed and then divided by the total number of pixels. The line statistics graph displays the general trend of DTest_mal mistakes through the blue line, while the yellow line reflects the overall trend of DTest_benign errors. The inherent unpredictability of the dataset plus the redundant nature of the software files result in a non-zero error. Figure 5 illustrates a significant disparity in the average error values between the two datasets. The blue line represents a consistently steady error trend for the malware dataset, while the yellow line represents an erratic and fluctuating error trend for the benign software test set. This supports our perspective.

Based on this experiment, we can show that the automated encoder can accurately detect complex characteristics of both harmless and harmful software. It can successfully rebuild the pre-processed malware data from our dataset. Next, we proceed to carry out the task of differentiating between harmful and benign software.

A. Performance Metrics Comparison

The performance of the autoencoder model is assessed using a range of metrics, including accuracy, precision, recall, F1-score, false positive rate, and false negative rate.

These metrics provide a comprehensive view of the model's ability to correctly identify malware instances while minimizing false positives and false negatives. To demonstrate the possible benefits of the autoencoder method, the outcomes are contrasted with conventional signature-based approaches. Malware that may change its structure or behavior can easily evade signature-based approaches because they are based on predetermined patterns. The autoencoder is a more adaptive approach since it can learn from the data's intrinsic properties without fixed signatures.

The ROC curves depicted in Figure 6 illustrate the impact of the model on the training set. It is evident that the model demonstrates a consistently reliable performance on the training set. Figure 7 illustrates the test set's performance of the model, namely Dataset-2, using ROC curves. Our model outperforms the other two models. The Datasets-3 are employed as AE-2 test sets to further examine the detection efficacy of our model on previously unidentified malware. Based on the ROC curves presented in Figure 8, our model exhibits high accuracy and shows promise in identifying previously unidentified malware. Nevertheless, it also demonstrates notable constraints as a result of the program's alteration with each iteration.

Figure 9 depicts the quantitative measures of accuracy, precision, recall, and F-score for the five models. Traditional machine learning (ML) detection techniques, such as decision trees, demonstrate superior accuracy and outperform naive Bayes and support vector machines in cross-sectional comparisons. However, for overall performance, deep learning models are the preferred choice. Our model surpasses the competitors in terms of search accuracy and completeness. The performance of the two deep learning models is compared in Figure 10. The chart demonstrates that AE-2 has a shorter training time compared to the CNN-0 model, although all the parameters, including ACC, recall, Precision, and F-score, are nearly the same. AE-2 exhibits a lower value for FPR.

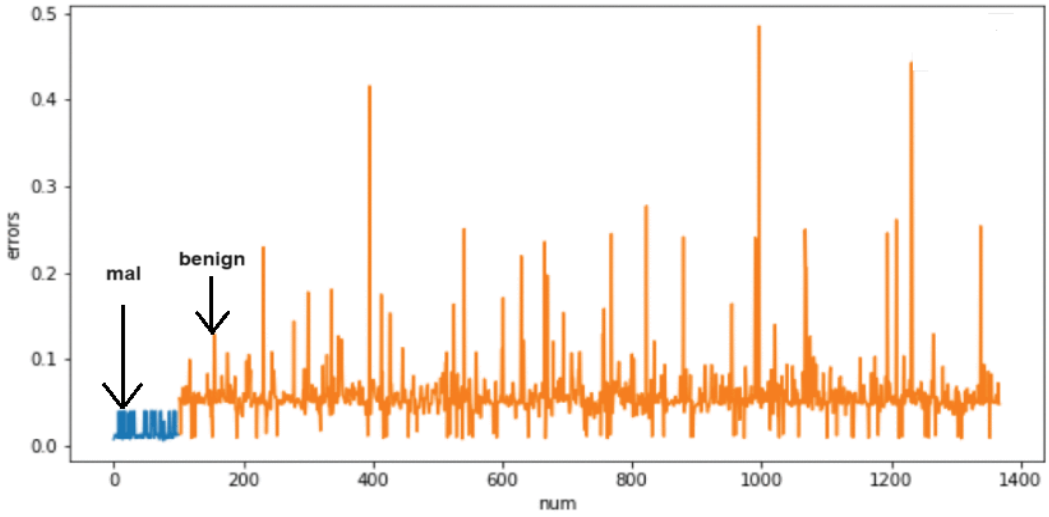

Figure 5. Mistake in reconstruction for two sets of data.

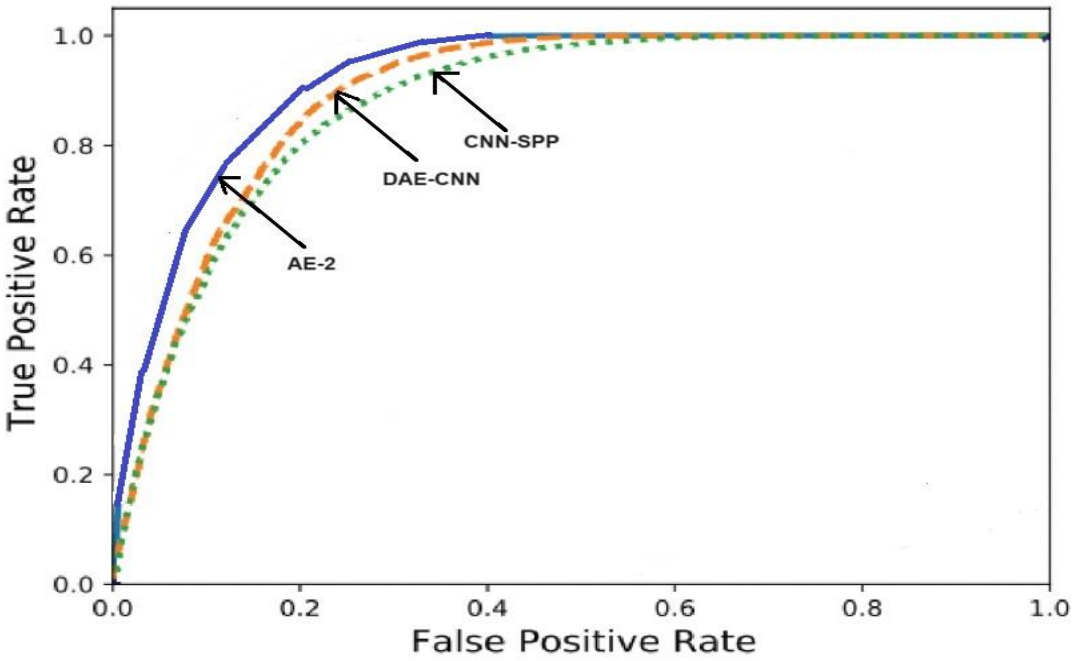

Figure 6. Shows the AE-2 ROC curve on the training set

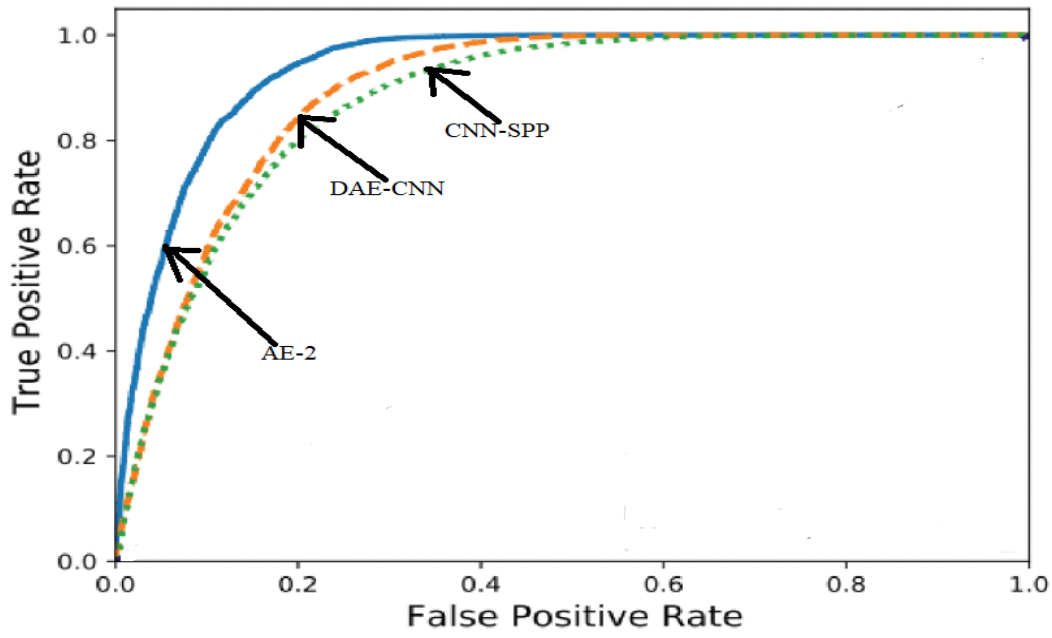

Figure7: ROC curves for several models on the validation data.

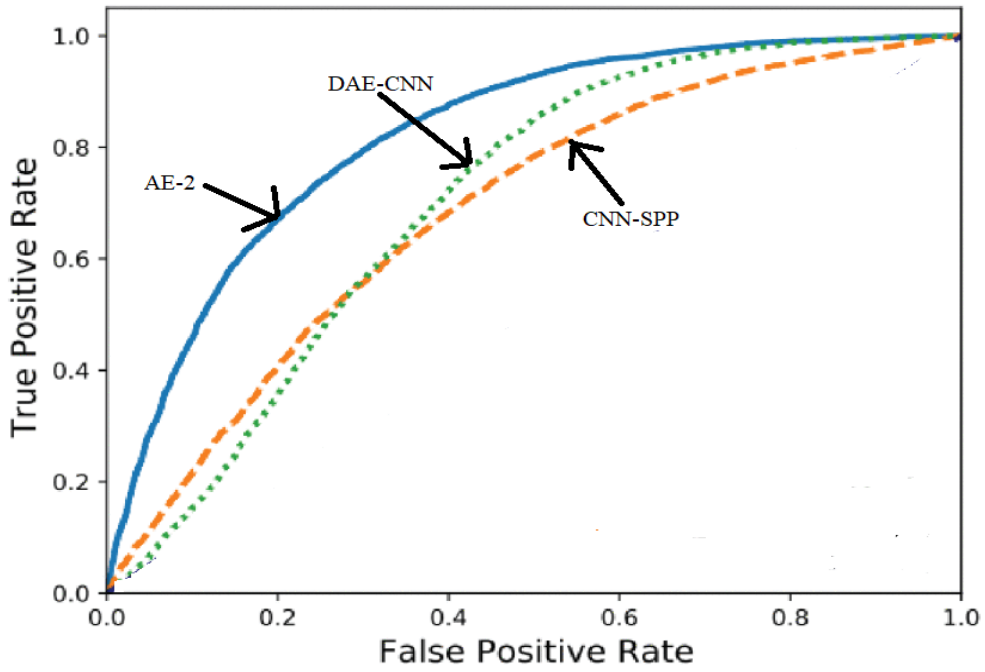

Figure 8. The ROC curve of many models on the unknown software

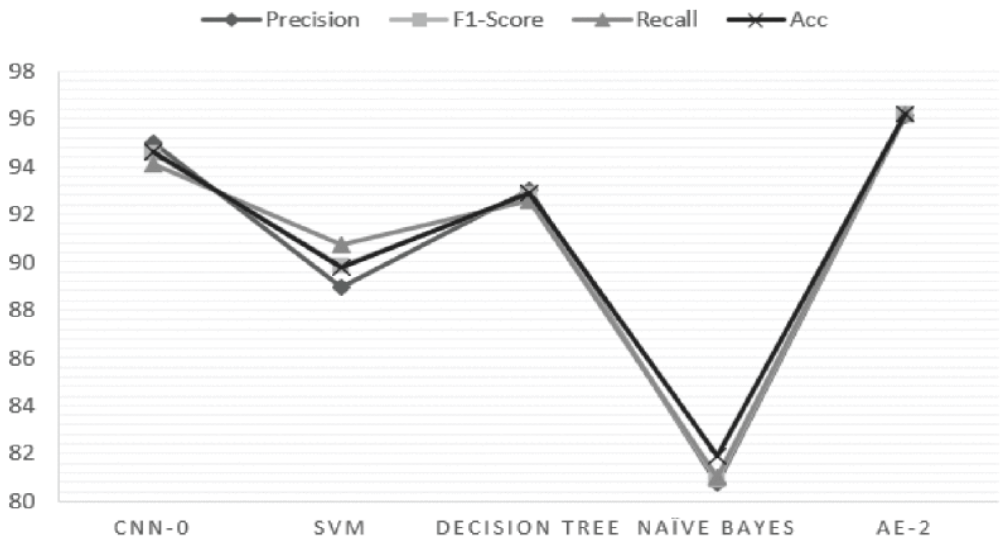

Figure 9. Shows the outcomes of comparing five distinct models

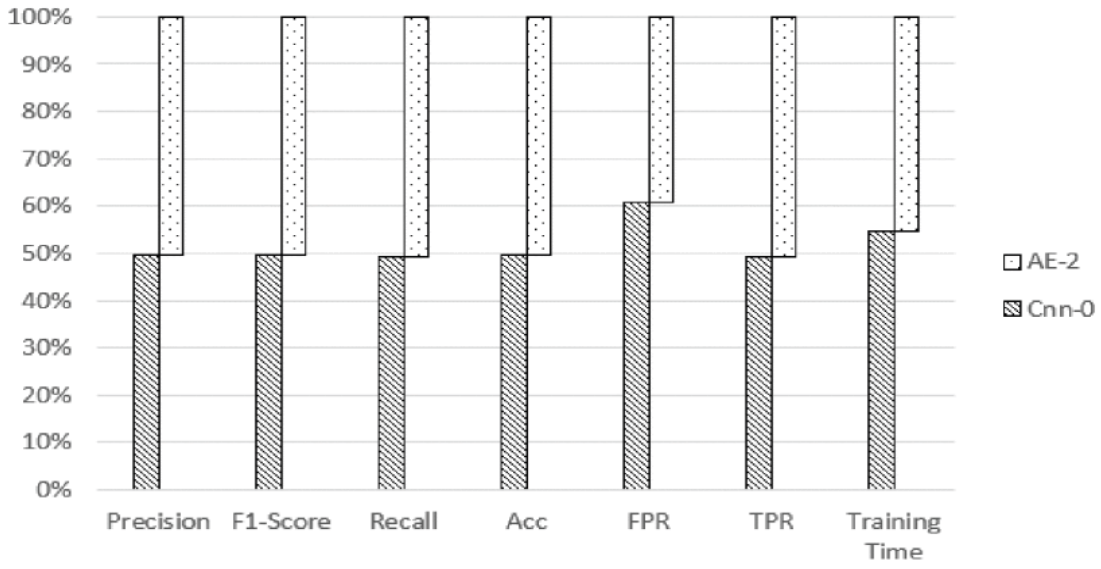

Figure 10. Two deep learning models' performance compared

The complete set of experimental data is shown in Table 1. The total time required to train our model is around 23.45 hours, or 1407.32 minutes. Due to the high number of parameters in the CNN-0 model, the picture data is processed 28.14 hours following the convolution and pooling methods. This is because there is just one convolutional layer and one pooling layer in the model's architecture.

TABLE 1
COMPREHENSIVE ANALYSIS OF SEVERAL INDICATIONS

Classifier	FPR %	TPR%	ACC %	Precision %	Recall %	Training time (minutes)
CNN-0	5.8	93.9	93.9	93.8	93.6	1699.4
SVM	10.9	91.1	90.2	90.0	89.1	2.01
Decision Tree	7.89	91.6	91.8	92.9	91.9	0.59
Naïve Bayes	19.0	80.9	82.1	81.1	79.9	1.67
AE-2	4.1	95.9	95.8	95.2	95.1	1389.4

V. CONCLUSION

The results of this research underscore the potential of auto encoders in advancing malware detection capabilities. By embracing the dynamic and unsupervised learning paradigm, auto encoders offer a promising avenue for enhancing the adaptability and effectiveness of cybersecurity defenses in the ongoing battle against evolving malware threats. As the digital landscape continues to evolve, the insights gained from this research contribute to the ongoing pursuit of innovative and resilient solutions to safeguard digital ecosystems. The experimental outcomes validate the viability of our suggested methodology, which involves transforming the bytecode of every software method into a grayscale image that visually represents the attributes of a software sample. Our method is significantly more precise in detecting malware than those developed using conventional machine learning algorithms. Our approach demonstrates reduced training and detection times in comparison to alternative malware detection systems that rely on deep learning models. Suggestions for future research directions are outlined, such as exploring ensemble methods combining autoencoders with other deep learning architectures, incorporating temporal aspects for dynamic malware detection, or leveraging adversarial training to enhance model robustness. These recommendations aim to guide subsequent investigations in the ongoing quest for more effective and adaptive malware detection systems.

REFERENCES

[1] Dhoni, Pan & Kumar, Ravinder. (2023). Synergizing Generative AI and Cybersecurity: Roles of Generative AI Entities, Companies, Agencies, and Government in Enhancing Cybersecurity. 10.36227/techrxiv. 23968809.v1.

[2] Demertzi, V., Demertzis, S., & Demertzis, K. (2022). An Overview of Cyber Threats, Attacks and Countermeasures on the Primary Domains of Smart Cities. Applied Sciences, 13(2), 790. https://doi.org/10.3390/app13020790

[3] Danial Javaheri, Mahdi Fahmideh, Hassan Chizari, Pooia Lalbakhsh, Junbeom Hur, "Cybersecurity threats in FinTech: A systematic review," Expert Systems with Applications, Volume 241, 2024.

[4] Xing, Xiaofei & Jin, Xiang & Elahi, Haroon & Jiang, Hai & Wang, Guojun. (2022). A Malware Detection Approach Using Autoencoder in Deep Learning. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3155695.

[5] Jin, Xiang & Xing, Xiaofei & Elahi, Haroon & Wang, Guojun & Jiang, Hai. (2020). A Malware Detection Approach Using Malware Images and Autoencoders. 1-6. 10.1109/MASS50613.2020.00009.

[6] D'Angelo, Gianni & Ficco, M. & Palmieri, Francesco. (2019). Malware detection in mobile environments based on Autoencoders and API-images. Journal of Parallel and Distributed Computing. 137. 10.1016/j.jpdc.2019.11.001.

[7] Kebede, Temesguen & Djaneye-Boundjou, Ouboti & Narayanan, Barath & Ralescu, Anca & Kapp, David. (2017). Classification of Malware programs using autoencoders based deep learning architecture and its application to the microsoft malware Classification challenge (BIG 2015) dataset. 70-75. 10.1109/NAECON.2017.8268747.

[8] Brezinski, Kenneth & Ferens, K. (2021). Metamorphic Malware and Obfuscation -A Survey of Techniques, Variants and Generation Kits. 10.13140/RG.2.2.19702.52802.

[9] Zahoor-Ur Rehman, Sidra Nasim Khan, Khan Muhammad, Jong Weon Lee, Zhihan Lv, Sung Wook Baik, Peer Azmat Shah, Khalid Awan, Irfan Mehmood, "Machine learning-assisted signature and heuristic-based detection of malwares in Android devices," Computers & Electrical Engineering, Volume 69, 2018.

[10] Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," in IEEE Access, vol. 8, pp. 6249-6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[11] N. Pachhala, S. Jothilakshmi and B. P. Battula, "A Comprehensive Survey on Identification of Malware Types and Malware Classification Using Machine Learning Techniques," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021, pp. 1207-1214, doi: 10.1109/ICOSEC51865.2021.9591763.

[12] Ahmed, S.F., Alam, M.S.B., Hassan, M. et al. Deep learning modelling techniques: current progress, applications, advantages, and challenges. Artif Intell Rev 56, 13521–13617 (2023). https://doi.org/10.1007/s10462-023-10466-8

[13] Abdel-Jaber, Hussein, Disha Devassy, Azhar Al Salam, Lamya Hidaytallah, and Malak EL-Amir. 2022. "A Review of Deep Learning Algorithms and Their Applications in Healthcare" Algorithms 15, no. 2: 71. https://doi.org/10.3390/a15020071

[14] Wasim Khan, Mohammad Haroon, "An unsupervised deep learning ensemble model for anomaly detection in static attributed social networks," International Journal of Cognitive Computing in Engineering, Volume 3, 2022.

[15] Sepehr Maleki, Sasan Maleki, Nicholas R. Jennings, "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering," Applied Soft Computing, Volume 108, 2021.

[16] Maleki, Sepehr & Maleki, Sasan & Jennings, Nicholas. (2021). Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering. Applied Soft Computing. 108. 107443. 10.1016/j.asoc.2021.107443.

[17] Mobtahej, Pooyan & Zhang, Xulong & Hamidi, Maryam & Zhang, Jing. (2022). An LSTM-Autoencoder Architecture for Anomaly Detection Applied on Compressors Audio Data. Computational and Mathematical Methods. 2022. 1-22. 10.1155/2022/3622426.

[18] Wang, Weiping & Wang, Zhaorong & Zhou, Zhanfan & Deng, Haixia & Zhao, Weiliang & Wang, Chunyang & Guo, Yongzhen. (2021). Anomaly detection of industrial control systems based on transfer learning. Tsinghua Science and Technology. 26. 821-832. 10.26599/TST.2020.9010041.

[19] Yuan, Xiaoyong. (2017). PhD Forum: Deep Learning-Based Real-Time Malware Detection with Multi-Stage Analysis. 1-2. 10.1109/SMARTCOMP.2017.7946997.

[20] Tayyab, Umm-e-Hani, Faiza Babar Khan, Muhammad Hanif Durad, Asifullah Khan, and Yeon Soo Lee. 2022. "A Survey of the Recent Trends in Deep Learning Based Malware Detection" Journal of Cybersecurity and Privacy 2, no. 4: 800-829. https://doi.org/10.3390/jcp2040041

[21] Sewak, Mohit & Sahay, Sanjay & Rathore, Hemant. (2018). An investigation of a deep learning based malware detection system. 1-5. 10.1145/3230833.3230835.

[22] Xing, Xiaofei & Jin, Xiang & Elahi, Haroon & Jiang, Hai & Wang, Guojun. (2022). A Malware Detection Approach Using Autoencoder in Deep Learning. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3155695.

[23] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications,

future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[24] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[25] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[26] Antunes, Mário & Oliveira, Luís & Seguro, Afonso & Veríssimo, João & Salgado, Ruben & Murteira, Tiago. (2022). Benchmarking Deep Learning Methods for Behaviour-Based Network Intrusion Detection. Informatics. 9. 29. 10.3390/informatics9010029.