Scientific Research Publishing

Open Journal of Applied Sciences, 2023, 13, 2480-2491
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Malware Detection Using Deep Learning

Achi Harrisson Thiziers, Koné Tiémoman, N'guessan Behou Gérard, Traoré Tiémoko Qouddouss Kabir

Unité de Recherche et d'Expertise Numérique (UREN), Université Virtuelle de Côte d'Ivoire, Abidjan, Côte d'Ivoire
Email: thiziers.achi@uvci.edu.ci, dg@uvci.edu.ci, behou.nguessan@uvci.edu.ci, tiemoko.traore@uvci.edu.ci

How to cite this paper: Thiziers, A.H., Tiémoman, K., Gérard, N.B. and Kabir, T.T.Q. (2023) Malware Detection Using Deep Learning. *Open Journal of Applied Sciences*, **13**, 2480-2491.
https://doi.org/10.4236/ojapps.2023.1312193

Received: November 28, 2023
Accepted: December 26, 2023
Published: December 29, 2023

Abstract

Malware represents a real threat to information systems, because of the damage it causes. This threat is growing today, as these programs take on more complex forms. This means they escape traditional malware detection methods. Hence the need for artificial intelligence, more specifically Deep Learning, which could detect malware more effectively. In this article, we've proposed a model for malware detection using artificial neural networks. Our approach used data from the characteristics of machines, particularly computers, to train our Deep Learning algorithm. This model demonstrated an accuracy of around 83% in predicting the presence of malware on a machine. Thus, the use of artificial neural networks for malware detection has shown his ability to assimilate complex, non-linear patterns from data.

Keywords

Neural Network, ANNs, Malicious Code, Malware Analysis, Artificial Intelligence

1. Introduction

With the rapid development of the Internet, malware has become one of today's major cyberthreats. As the diversity of malware increases, antivirus software is no longer able to meet security needs. As a result, millions of computers are under attack. According to Kaspersky Labs (2016), 6,563,145 different hosts were attacked, and 4,000,000 unique malicious objects were detected in 2015. While traditional methods, such as signature-based static analysis, have been foundational in combating known threats, the dynamic and sophisticated nature of modern malware necessitates more advanced detection strategies. This paradigm shift towards deep learning in malware detection has led to substantial improvements in identifying previously unseen and polymorphic malware variants.

However, despite the progress made, there exists a research gap in understanding the robustness, scalability, and generalizability of deep learning models across diverse malware landscapes. By focusing on this crucial research gap, the proposed study seeks to contribute valuable knowledge to the existing literature, providing a nuanced understanding of the capabilities and challenges associated with deploying deep learning for malware detection. Through empirical evaluations and rigorous experimentation, the research endeavors to offer practical recommendations for optimizing deep learning models, thereby fostering the development of more robust and resilient malware detection systems in the face of an ever-evolving cyber threat landscape. This paper presents elements of solutions for detecting malware based on artificial intelligence, mainly Deep Learning.

2. Malware Basics

2.1. Malware Definition

Malware is a generic term for any type of computer program or software known as "malicious software". It covers a range of techniques and tools designed to infect and damage their victims' hardware, silently exploit their resources, extort money and/or steal sensitive and important data. They are used by cybercriminals for financial gain. There are different types of malwares, the best known of which are: Ransomware (Bounaamane, B., & Drissi, Z. (2023)) [1], Trojans (Moffie, *et al.* 2006) [2], Spyware, Backdoor, Viruses (Horton and Seberry 1997) [3], Remote Administration Tools (RAT), Worms, Rootkits (Chuvakin 2003) [4].

2.2. Malware Detection Methods

All malware detection techniques can be divided into two categories: static methods based on signatures; dynamic methods based on behavior [5].

2.2.1. Static Methods Based on Signatures

Signature-based analysis is a static method based on predefined signatures. These can be file fingerprints, such as MD5 or SHA1 hashes, static strings, or file metadata. The detection scenario, in this case, would be as follows: when a file arrives on the system, it is statically analyzed by the antivirus software. If one of the signatures matches, an alert is triggered, indicating that the file is suspicious [6].

2.2.2. Dynamic Methods Based on Behavior (Behavior-Based Analysis)

Today's attackers are developing malware that can modify its signature. This characteristic of malware is called polymorphism. Antivirus vendors have therefore had to find another means of detection, based on behavior or heuristics. In this method, the actual behavior of malware is observed during execution, looking for signs of malicious behavior: modification of host files, registry keys, establishment of suspicious connections [7].

2.2.3. Some Signs of Malware in the Information System

The daily impact of malware is disabling your computers and technological tools; Data loss and theft; Financial losses. Some signs of malware infection are computer slowness; problems shutting down or starting up your computer (slow startup); Infection warnings, often accompanied by purchase requests to remedy the situation; presence of unknown programs on your computer. Figure 1 illustrates the presence of malware in the system.

2.3. The Need for Artificial Intelligence and Deep Learning

The two previous methods are increasingly outdated by the complexity of malware, and a solution based on artificial intelligence has been considered by several studies.

Recent work by Ijaz, Durad & Ismail in 2019 focuses on modern malware, which is represented by its runtime behaviors and collected using the Cuckoo Sandbox tool. The authors demonstrate that better predictive accuracy is achieved with static analysis, due to the predominance of simple malware. However, they point out that dynamic analysis is preferable due to the protection mechanisms used by malware [8].

Kolosnjaji, Zarras, Webster & Eckert in 2016 applied neural networks to old malware in PE format (32-bit Windows executables). Although the results obtained were interesting, this methodology cannot be implemented in real time due to the need for post-processing to remove repeated actions [9].

3. Material and Methods of Our Proposed Model

3.1. Material of Our Proposed Model

We used Microsoft Malware Prediction dataset, from Kaggle database, to build our prediction models. "Microsoft Malware Prediction" is a set of binary classification data (HasDetections "0" or "1"), containing around 16,774,736 machines and 81 attributes (machine and target identifier "HasDetections"), including 52 categorical attributes (23 attributes are attributes are numerically

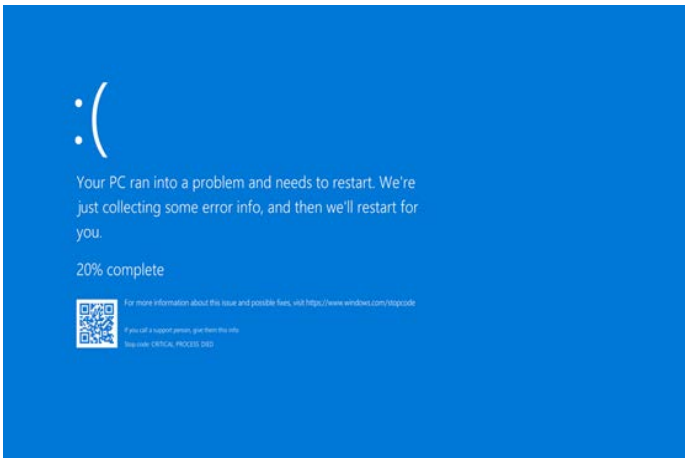

Figure 1. The blue screen of a computer apparently infected by malware.

coded to protect data confidentiality). Our data source consists of 2 csv files, one of which is the training file (train.csv) and the other the test file (test.csv). Together, these files contain 83 columns or attributes describing various information concerning the hardware and software of each machine. Of these, 53 are numerical values. The train.csv file contains 567,730 machines and 83 attributes, while the test.csv contains 243,313 machines with 82 columns. The overview of our dataset is illustrated by the two images (see Figure 2 and Figure 3).

3.2. Description of the Dataset

The dataset used is made up mainly of healthy machines, as shown by the distribution of data according to labels, as shown on Figure 4.

3.3. Method of Our Proposed Model

3.3.1. Data Pre-Processing

The first step consisted of dealing with missing values and outliers, the analysis of the dataset has revealed that more than 44 columns have missing values, and between them around 7 columns have more than 50% missing data. The visual representation below shows how missing values are distributed before data processing (see Figure 5).

The second step consisted of data balancing according to the targeted column. Analyzing the target column, we observe an imbalance of occurrences, namely: 482,571 machines not infected by malware; 85,159 malware-infected machines.

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion	IsBeta	RtpStateBitfield	IsSxsPassiveMode	DefaultBrowser
0	69cb692a4989148b1f1d02cbb3453688	win8defender	1.1.15200.1	4.18.1807.18075	1.275.1320.0	0	7.0	0	
1	521257fc4824d66dd7b02a3bc4107b1c	win8defender	1.1.14202.0	4.13.17134.228	1.253.125.0	0	7.0	0	
2	537bd077fecd049b7da4e8f4cc849fec	win8defender	1.1.15200.1	4.8.10240.17443	1.275.1519.0	0	7.0	0	
3	61e4980f491dd2c13709b361bbc5e508	win8defender	1.1.15000.2	4.18.1806.18062	1.271.1124.0	0	7.0	0	
4	5ae9cab8a552c0159246b208a08e9b47	win8defender	1.1.15100.1	4.16.17656.18052	1.273.1056.0	0	7.0	0	
5	588892960aa83150aee42f088c438eeb	win8defender	1.1.15200.1	4.8.10240.17443	1.275.1120.0	0	7.0	0	
6	cd3540eb18c73f1c229f5c259cb307aa	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1112.0	0	7.0	0	
7	5d717fc122d243703f2b8b7ab7ee91da	win8defender	1.1.15100.1	4.18.1806.18062	1.273.483.0	0	7.0	0	
8	d2e3a2c36e0026fb210bd13c4109a2b8	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1699.0	0	7.0	0	
9	69978c06d1b4d2c14fc13d4b781d873f	win8defender	1.1.15000.2	4.18.1806.18062	1.271.1162.0	0	7.0	0	

Figure 2. Training file overview (train.csv).

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion	IsBeta	RtpStateBitfield	IsSxsPassiveMode	DefaultBrowser
0	69cb692a4989148b1f1d02cbb3453688	win8defender	1.1.15200.1	4.18.1807.18075	1.275.1320.0	0	7.0	0	
1	521257fc4824d66dd7b02a3bc4107b1c	win8defender	1.1.14202.0	4.13.17134.228	1.253.125.0	0	7.0	0	
2	537bd077fecd049b7da4e8f4cc849fec	win8defender	1.1.15200.1	4.8.10240.17443	1.275.1519.0	0	7.0	0	
3	61e4980f491dd2c13709b361bbc5e508	win8defender	1.1.15000.2	4.18.1806.18062	1.271.1124.0	0	7.0	0	
4	5ae9cab8a552c0159246b208a08e9b47	win8defender	1.1.15100.1	4.16.17656.18052	1.273.1056.0	0	7.0	0	
5	588892960aa83150aee42f088c438eeb	win8defender	1.1.15200.1	4.8.10240.17443	1.275.1120.0	0	7.0	0	
6	cd3540eb18c73f1c229f5c259cb307aa	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1112.0	0	7.0	0	
7	5d717fc122d243703f2b8b7ab7ee91da	win8defender	1.1.15100.1	4.18.1806.18062	1.273.483.0	0	7.0	0	
8	d2e3a2c36e0026fb210bd13c4109a2b8	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1699.0	0	7.0	0	
9	69978c06d1b4d2c14fc13d4b781d873f	win8defender	1.1.15000.2	4.18.1806.18062	1.271.1162.0	0	7.0	0	

Figure 3. Test file overview (train.csv).

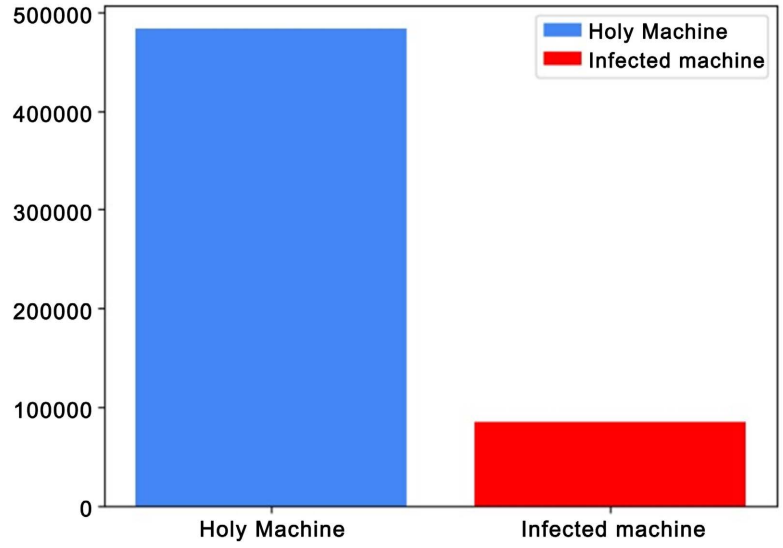

Figure 4. Target column visualization (HasDetections).

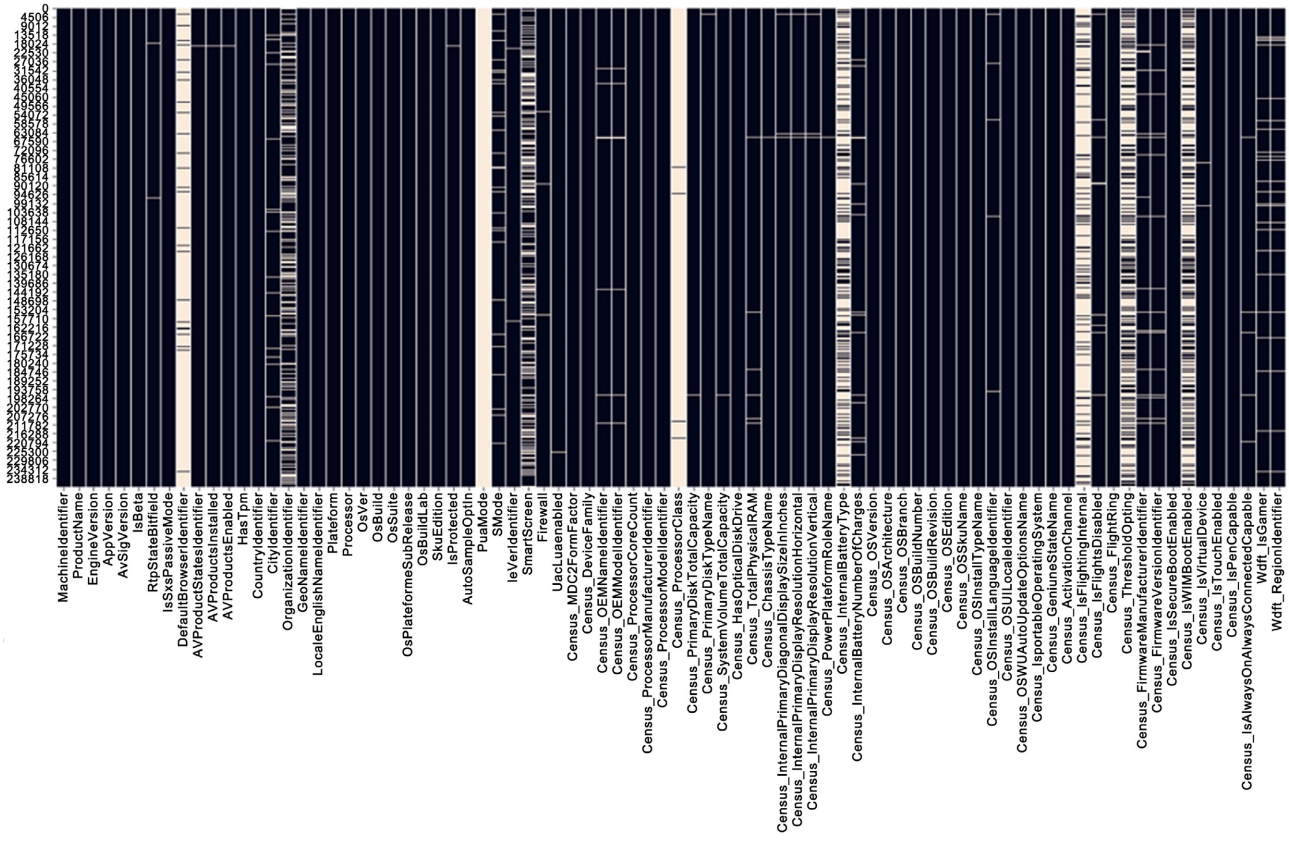

Figure 5. Distribution of missing values by column.

For better learning, we're going to balance this out by putting them at the same level, as shown in the figure below. At the end of this resampling, we obtain a data set containing as many healthy machines as infected machines (see **Figure 6**).

The following **Figure 7** shows the distribution of infected and uninfected machines after data preprocessing.

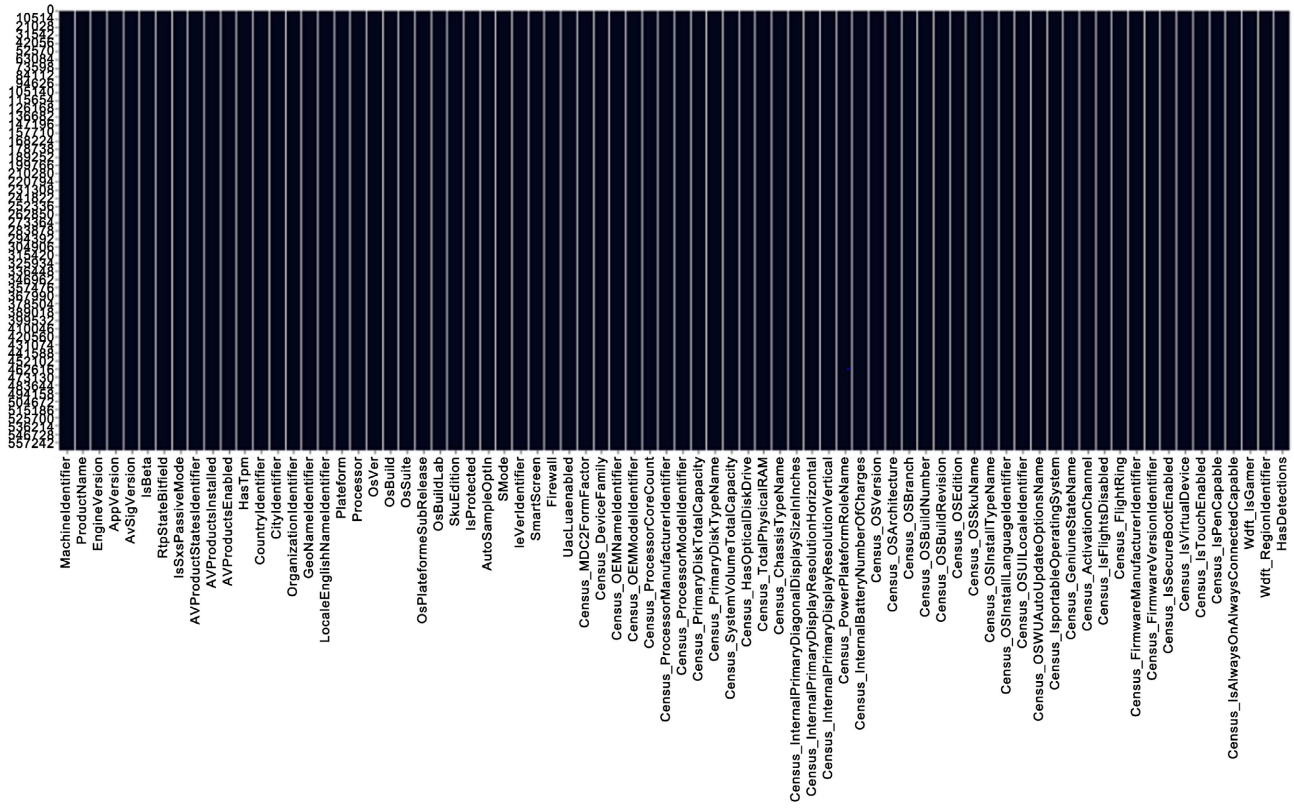

Figure 6. Distribution of missing values after processing.

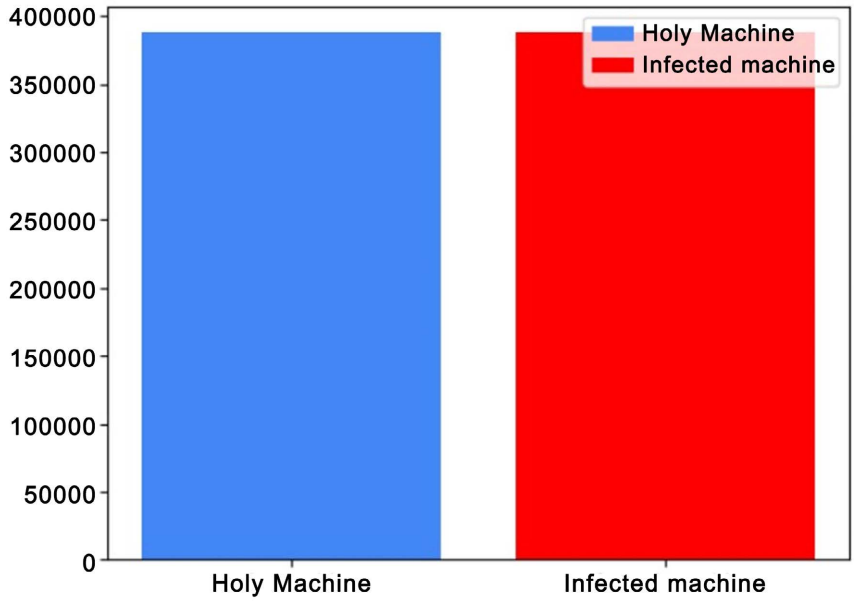

Figure 7. Distribution of healthy and infected machines after resampling.

The last step is encoding. We used the following policy: for any categorical variable with more than 2 unique categories, we used label encoding. As shown in the illustration of Figure 8, columns with categorical values are encoded using the LabelEncoder function.

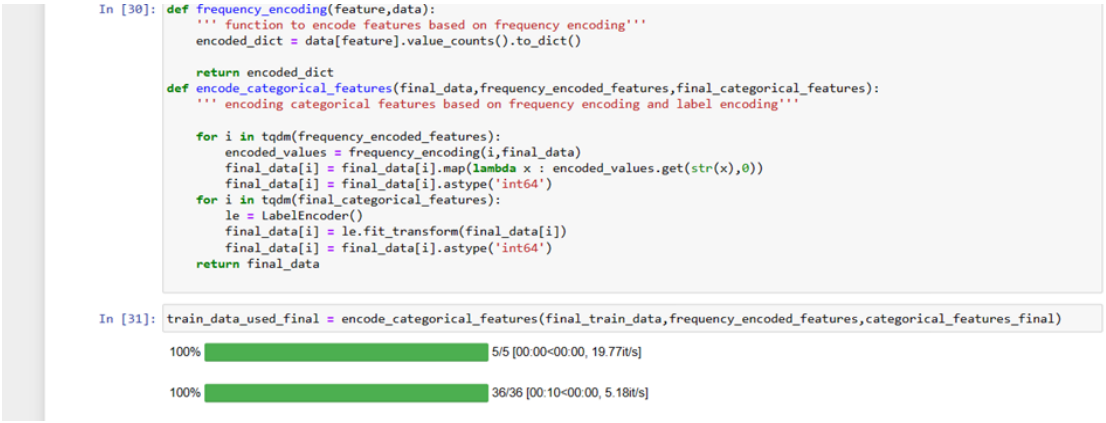

Figure 8. Encoding categorical values.

3.3.2. Development Environment

The model was deployed in a python environment, on a HP Gamer Omen RTX computer, 64 GB Ram, Core i9, 2 Tera SSD, with the following tools: Python 3.9; Jupyter Notebook; Google Colab; These are the following libraries: Matplotlib; Numpy; Pandas; Seaborn; Scikit-learn, TensorFlow, Keras. We used 70% of our dataset to train our neural network, while we used 30% of our data for testing.

3.3.3. Our Deep Learning Algorithm Used: Artificial Neural Network (ANN)

To implement our artificial neural network (ANN), we went through several steps: data pre-processing, data scaling and finally data partitioning into training and test samples. The data partitioning is illustrated in the figure below (see **Figure 9**).

Our ANN, based on a biological model, is represented as follows, with each ball representing a neuron (see **Figure 10**).

- The set $\{w_0, w_1, \ldots, w_n\}$ represents the weights associated with each neuron
- The set $\{x_0, x_1, \ldots, x_n\}$ represents the input values for each neuron
- f represents the activation function
- b represents the neuron's bias
- The output s of a neuron in each layer is calculated by applying the activation function f to the sum of the input values x_k weighted by the weights w_k and summed with the bias b.

3.3.4. Our ANN Architecture

The model used for our study is an artificial neural network (ANN) whose modeling approach is described in the diagram of **Figure 11**.

4. Results and Discussion

4.1. Results

To evaluate the performance of our model, we relied on certain classification evaluation measures, namely:

- The Matrix confusion [10] is shown on **Figure 12**

```
In [34]: X= new_train_data.drop(['HasDetections'], axis=1)
         y= new_train_data['HasDetections']

In [35]: from sklearn.preprocessing import MinMaxScaler, StandardScaler

         scaler = StandardScaler()
         X_scale = scaler.fit_transform(X)
         X_use = X_scale

In [37]: X_train, X_test, y_train, y_test = train_test_split(X_use, y, test_size=0.3, random_state=42)
```

Figure 9. Illustration of data separation.

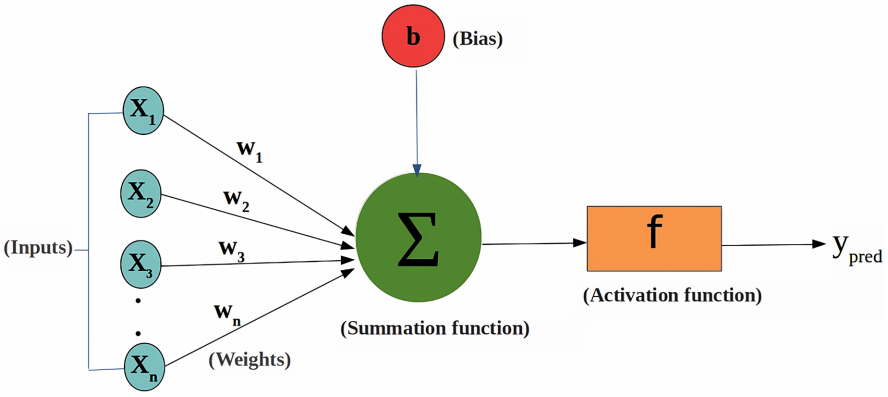

Figure 10. Illustration of the calculation of a neuron's output value.

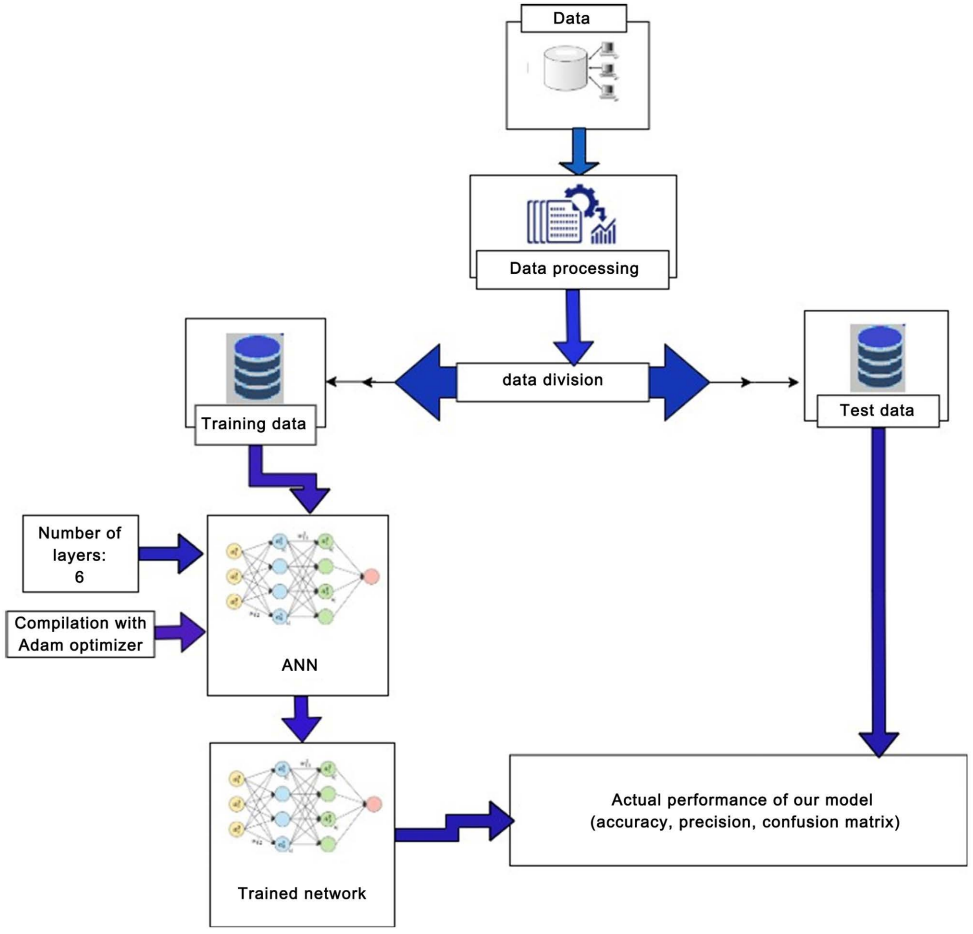

Figure 11. Modeling our ANN.

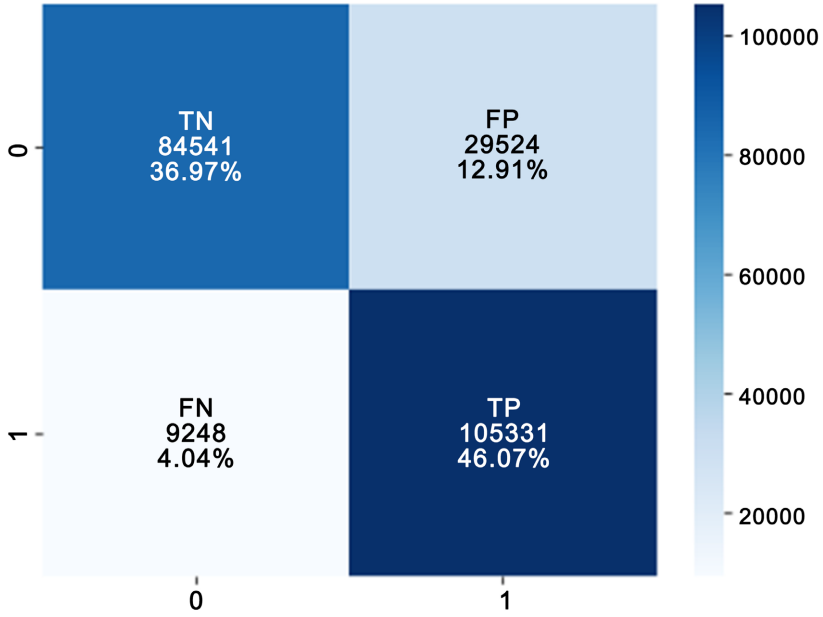

Figure 12. Model confusion matrix.

- TP: Our model predicts 84,541 true positives
- TN: Our model predicts that 29,524 machines are not infected, but are in fact infected
- FP: Our model predicts that 105,331 machines will be infected
- FN: Our model predicts that 9248 machines are infected, but they are not.
- Precision score

We trained an Artificial Neural Network (ANN) using Keras to predict the state of machines in relation to malware. Our model achieved an accuracy rate of 83.04%, as shown in **Figure 13**.

The score was obtained thanks to good training of the neural network. This training gave the following performance (see **Figure 14**).

On the left-hand side, the error rate decreases, while the accuracy increases over time. The error rate has fallen from around 60% to 20%, demonstrating the accuracy of the model.

- The classification report:

- Precision: Precision is the proportion of instances correctly classified for a given class among all instances predicted to belong to that class.
- Recall: Recall, also known as sensitivity, measures the proportion of instances correctly classified for a given class among all instances belonging to that class.
- F1-score: F1-score is a metric that combines both precision and recall into a single value to assess prediction accuracy. It is particularly useful when classes are unbalanced.
- Support: Support represents the number of real instances for each class, allowing us to evaluate the distribution of the data. We can see all these in **Figure 15**.

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import accuracy_score
# from keras.utils import np_utils
import itertools

accuracy_score(y_test, ynew)
```

```
0.8304263396371652
```

Figure 13. Precision score.

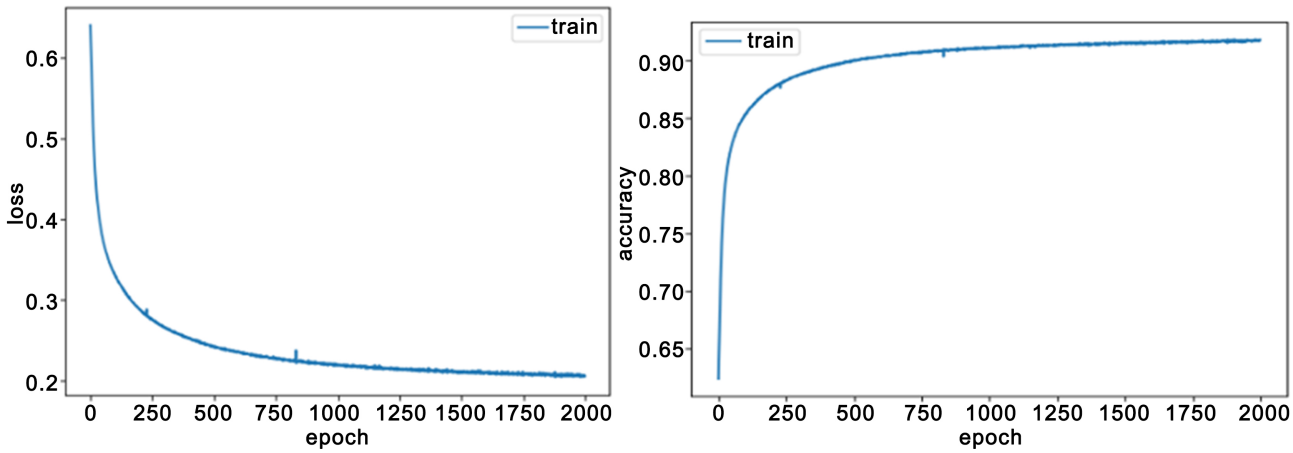

Figure 14. Model performance.

	precision	recall	f1-score	support
0	0.90	0.74	0.81	114065
1	0.78	0.92	0.84	114579
accuracy			0.83	228644
macro avg	0.84	0.83	0.83	228644
weighted avg	0.84	0.83	0.83	228644

Figure 15. The classification report of the test.

4.2. Discussion

The solution we have proposed for detecting malware within machines or computers enables us to identify the presence of malware. However, the validity of our study may vary depending on whether it is applied to machines equipped with non-proprietary operating systems, such as open-source ones. Indeed, our research has mainly focused on machines running the Windows operating system. Our approach emphasizes machine characteristics, whereas existing studies focus more on the functional aspects of malware. Consequently, extending our findings to open-source environments would require in-depth analysis to assess the feasibility of applying our method.

4.3. Comparison

Given the impact that malware can have on information systems, malware de-

tection is all the more important in cybersecurity.

Z. Zhang, in an article published in 2022, used LightGBM, a machine learning model. His work was based on the same dataset as ours. LightGBM is a boosting model based on the gradient boosting algorithm, designed to improve performance and efficiency over other boosting methods such as XGBoost or Gradient Boosting Machine (GBM). Compared with our work, it scored below ours, which is 0.684 [11].

D. Huo and colleagues have also carried out research into software detection. They used two learning methods, namely LightGBM and a one-dimensional CNN, with respective scores of 67.16% and 72.47% [12]. In the following **Table 1**, we have grouped together the different scores achieved by previous works, compared with the results of our own.

Table 1. Model comparison table.

Model	Score (%)	Authors
LightGBM	68.40	Z. Zhang
CNN unidimensionnel 1D	72.47	D. Huo, X. Li, L. Li, Y. Gao, X. Li and J. Yuan
LightGBM*	67.16	D. Huo, X. Li, L. Li, Y. Gao, X. Li and J. Yuan
Our ANN	83.00	

5. Conclusion

In this work, we have proposed and implemented a model for detecting malware within machines. This is a deep learning model based on artificial neural networks, to provide a highly efficient model. This work has enabled us to develop a malware detection model using data derived from machine characteristics. The python programming language and TensorFlow library were used to implement this model. The resulting model, based on neural networks, can predict the presence or absence of malware with an accuracy of 83%. The use of this model will enabled malware to be detected effectively, even if it could modify code to evade traditional detection methods and avoid attacks at an early stage.

Acknowledgements

We would like to thank the Université Virtuelle de Côte d'Ivoire for allowing us to carry out this research work.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Bounaamane, B. and Drissi, Z. (2023) Cybercriminalité et cyber-résilience: Les enjeux de la sécurité numérique dans un monde connecté. *International Journal of*

Accounting, *Finance*, *Auditing*, *Management and Economics*, **4**, 451-469.

[2] Moffie, M. and Cheng, W. (2006) Hunting Trojan Horses. *Proceedings of the* 1*st Workshop on Architectural and System Support for Improving Software Dependability*, San Jose, California, 21 October 2006, 12-17. https://doi.org/10.1145/1181309.1181312

[3] Horton, J. and Seberry, J. (1997) Computer Viruses: An Introduction. In: Patel. M., Ed., *Proceedings of the Twentieth Australasian Computer Science Conference* (*ACSC*'97), *Computer Science Communications*, Vol. 19, 122-131.

[4] Chuvakin, A. (2003, February) An Overview of Unix Rootkits. dans iALERT White Paper, iDefense Labs, Chantilly, VA, p. 27.

[5] Souri, A. and Hosseini, R. (2018) A State-of-the-Art Survey of Malware Detection Approaches Using Data Mining Techniques. *Human-Centric Computing and Information Sciences*, **8**, Article No. 3. https://doi.org/10.1186/s13673-018-0125-x

[6] Osho, O. and Hong, S. (2021) A Survey Paper on Machine Learning Approaches to Intrusion Detection. *International Journal of Engineering Research & Technology* (*IJERT*), **10**, 94-102. https://doi.org/10.17577/IJERTV10IS010040

[7] Almasoudy, F.H., Al-Yaseen, W.L. and Idrees, A.K. (2020) Differential Evolution Wrapper Feature Selection for Intrusion Detection System. *Procedia Computer Science*, **167**, 1230-1239. https://doi.org/10.1016/j.procs.2020.03.438

[8] Ijaz, M., Durad, M.H. and Ismail, M. (2019) Static and Dynamic Malware Analysis Using Machine Learning. 2019 16*th International Bhurban Conference on Applied Sciences and Technology* (*IBCAST*), Islamabad, 8-12 January 2019, 687-691. https://doi.org/10.1109/IBCAST.2019.8667136

[9] Kolosnjaji, B., Zarras, A., Webster, G. and Eckert, C. (2016) Deep Learning for Classification of Malware System Calls Sequences. In: AI 2016: *Advances in Artificial Intelligence*: 29*th Australasian Joint Conference*, Springer International Publishing, Cham, 137-149. https://doi.org/10.1007/978-3-319-50127-7_11

[10] https://en.wikipedia.org/wiki/Confusion_matrix

[11] Zhang, Z. (2022) Microsoft Malware Prediction Using LightGBM Model. 2022 3*rd International Conference on Big Data*, *Artificial Intelligence and Internet of Things Engineering* (*ICBAIE*), Xi'an, 15-17 July 2022, 41-44. https://doi.org/10.1109/ICBAIE56435.2022.9985850

[12] Huo, D., Li, X., Li, L., Gao, Y., Li, X. and Yuan, J. (2022) The Application of 1D-CNN in Microsoft Malware Detection. 2022 7*th International Conference on Big Data Analytics* (*ICBDA*), Guangzhou, 4-6 March 2022, 181-187. https://doi.org/10.1109/ICBDA55095.2022.9760349