Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 215 (2022) 148–157

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Innovative Data Communication Technology and Application

# A Novel Approach towards Windows Malware Detection System Using Deep Neural Networks

Usha Divakarla[a], K Hemant Kumar Reddy[b], K Chandrasekaran*

[a]*Information Science & Engineering, NMAMIT, NITTE, Karkala, India*
[b]*Computer Science & Engingerring, VIT AP University, Andhra Pradesh, India*
**Computer Science & Engineering, NITK, Surathkal, India*
ushachavali@gmail.com, khemant.reddy@gmail.com,kch@nitk.edu.in

## Abstract

Now-a-day's malicious software is increasing in numbers and at present becomes more harmful for any digital equipment like mobile, tablet, and computers. Traditional techniques like static and dynamic analysis, signature-based detection methods are become absolute and not effective at all. The advanced techniques like code encryption and code packing techniques can be used to hide detection; polymorphic malware is a new class of malware that changes their code structure from time to time to avoid detection, so there is a need for an intelligent system which can efficiently analyze the features of a new, unknown executable file and classify it correctly. There have been learning-based malware detection systems proposed in the literature, but most of those proposed approaches present a high accuracy over a small dataset, whereas the performance is very poor over industry-standard datasets. Operating system like windows is always in prime malware target because of the sheer high number of users. This paper proposes a simple, deep learning-based detection approachthat classifies a specified executable into benign or harmful. It has been trained using EMBER, an industry-level Windows malware dataset and tests with an accuracy of 87.76%.

© 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 4th International Conference on Innovative Data Communication Technologies and Application

*Keywords:*Malware, Windows, Operating System security, Static Analysis, Deep Learning, Neural Networks

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000 .
*E-mail address:*kch@nitk.edu.in

1877-0509 © 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 4th International Conference on Innovative Data Communication Technologies and Application
10.1016/j.procs.2022.12.017

## 1. Introduction

The problem of malware detection has been a long-time research problem in the security community. According to Kaspersky [1], it detects over 360,000 new malware samples every single day. There is a need to develop efficient methods to classify a new, unknown executable file as either into benign or harmful category. There are number of methods and techniques to detect existing known malware. But the actual problem lies in detecting a completely new, unknown piece of malwares. That is point where traditional methods like static analysis, dynamic analysis, and signature-based methods become ineffective for present scenarios. Static analysis and Dynamic analysis can help to a certain extent but because of the increasing complexity in the malware structure, they are rendering ineffective. Research community has been focusing on classifying these malwares using machine learning. This work presents a simple, deep learning-based malware detection system which classifies a given new, unknown Windows executable file as benign or harmful. The approach presented in this paper is an application of Deep Neural Networks. Deep Neural Networks is a machine learning technique where multiple hidden layers are present between input and output layers and these layers naturally increases the learning appetite of the system. It has been used to solve hard problems like image classification, text classification, and audio classification with extremely high accuracy. This work becomes witness that deep neural networks can be used to solve the malware detection problem with high accuracy in highly complex systems also.Now-a-days industry-level malware detection software, antivirus software are mostly cloud based software. The part of antivirus software present in the machine is an interface to detect new files in the system. It also contains preliminary detection techniques. But the main detection engine is present in the cloud. The engine is typically a machine learning model which has been trained frequently with variable time interval. The Approach presented in this work can be used as the main detection engine. It represents the core of any malware detection, anti-malware software and our work presents a simple deep learning model which is trained on EMBER[2], and an industry-level windows malware dataset with 1.1 million data samples which gives a test accuracy of 98.53%.The rest of the paper is as follows. Section 2 presents an extensive literature survey on various learning-based techniques used for malware detection. Section 3 presents the malware detection classifier, and its subsections presents in-depth explanation of the model and technique used, section 4 presents DLMD model with complete details, section 5 presents' experimental details and result discussion. Section 6 concludes the overall performance achievement of the proposed approach.

## 2. Literature Review

Malware is a piece of software that was written to harm data or devices where data resides. Malware industry has grown very rapidly in the past few years that organizations have invested heavily in technologies to evade traditional protection, forcing the anti-malware groups to build more robust software to detect and terminate the attacks. Machine learning is one technique that helps you to predict the malware in the system and thus helps you to take preventive action. This section has a gist of all the works relevant to the problem of malware detection and the usage of machine learning techniques to help in malware detection. Chandrasekar et al. 2012 [3] designed a system based on Windows API call sequencing and association mining. A 3rd order Markov chain was used to model the API call sequence and that feature set was provided to the association mining-based classifier which classified the given file as either benign or harmful. This system displayed a test accuracy of 90%. It outperformed all the previously existing intelligent detection systems. Filho et al. [4] proposed a malware detection system based on Objective Oriented Association mining. It is a rule-based classifier where the rules were by a variant of the Fast-FP Growth algorithm. Their work outperformed the accuracy and efficiency of the Naive Bayes, Support Vector Machines (SVM), Decision Tree based techniques used by Norton, McAfee and VirusScan. The system displayed a test accuracy of 64.14%.Lad et.al. [5] suggested a model capable of effectively categorising static PE files and extracting a feature set from the dataset. The purpose of the research was to highlight the significance of feature extraction rather than model creation. When given to neural networks with few layers, the well-extracted characteristics aid in producing superior results. Using the fewest number of layers will improve the model's performance and need less processing and testing time. Mane et.al. [6] proposed to discover the characteristics that best describe the given training data, deep neural networks (DNN) are used. In this case, a deep neural network learns the features from portable executable files. Deep neural network-based solutions are therefore efficient at detecting both known and novel malware while having low false positive rates. Li et.al. [7] proposed a deep learning-based malware detection framework. They executed a joint representation of several APIs to capture the programme behaviour by first applying embedding and convolutional layers. Second, by expressing the semantic data of each API call using the category, action, and operation objects of the API. Finally, they mined the relationship data between APIs using the Bi-LSTM module. Anil et.al. [8] are addressing the malware detection using decision tree (DT) and using C4.5 (J48) classification algorithm to find a suitable solution method for malware attack in a API routine.George et al. 2016 [9] introduced a large-scale malware classification system using random projections and neural networks. The novelty of this work is not the usage of neural networks, but the unique technique used to reduce the input dimensions by projecting each original input onto a vector with relatively smaller dimensions and use this projection as input to the model. A novel method of dimensionality reduction is seen here.William et al. 2016 [10]proposed a system which is a combination of two models: Unsupervised feature learning using stacked AutoEncoders(SAEs) and supervised learning (DNNs) for the actual parameter learning. The model is trained on a dataset with 50,000 binary samples and gives a test accuracy of 95.64%.Rushabh et al. 2017 [11] proposed a detection system which used an ensemble of machine learning models. It used K-Nearest Neighbors (KNN), Decision Tree, SVMs and Random Forest algorithms. It averages a test accuracy of 98.7%. The main drawback of this work is that it was trained on a small dataset consisting of 10,400 binaries. Ajit et al. 2017 [12] not only used an ensemble of machine learning algorithms to design the classification system, but they also used an improved feature set by synthesizing new features from the raw features extracted from the binary samples. A test accuracy of 98% was achieved when trained along with the new features. The main drawback of this work is that the system was trained on a very small dataset consisting of 5210 binary samples.Jeffrey 2017 [13] uses a Convolutional Neural Network (CNN) based system to detect malware. A binary is seen as an array of bytes analogous to how an image can be seen as an array of pixels. Bytes sequences of the strongest activations are identified. A test accuracy of 99.19% was obtained from this system which was trained with 0.4 million binary samples.

Mohammad et al. 2017 [14] have built classifiers using multiple learning algorithms like SVMs, Decision Tree and Naive Bayes. Nothing about the performance of these models have been mentioned in the paper.Zeliang et al. 2018 [15] used a Deep Convolutional Neural Network(CNN) and the raw inputs to the system are sequences of instructions grouped in a certain manner which are generated by their own tool. The model was trained on 70,000 windows binary samples and an overall accuracy of 95% was achieved. DNNs have been used to solve problems in a lot of diverse fields and they include fields outside computer science. It is extensively used in the medical and diagnostics field [16] to detect skin cancer [17], real time object detection [18] etc., used in object recognition and computer vision, Natural Language Processing (NLP). It has also been used in the field of computer security for spam detection [19], Malicious URL detection [20], Intrusion Detection System(IDS) [21]. Derivatives of DNNs like Generative Adversarial Networks(GANs) [22] are used to build offensive tools to attack various types of security software like Antivirus software[23], IDS [24]etc., Joshua et al. 2015 [25] proposed deep neural network based system. The model performs with a test accuracy of 95%. There are 3 reasons behind this system's success. Two-dimensional binary program features are used and fed to the deep learning model. The second reason is that the model is a deep neural network. It is a simple model with 2 hidden layers. Third is a large, real-world dataset with 400,000(0.4 million) binaries. All the methods have extracted features by analyzing binary samples statically. A lot of features go unnoticed during static analysis. Wei et al. 2018 [26] used dynamic analysis to generate the feature set and it is fed to a deep learning model. This model not only classifies the given binary as benign or harmful. If it is classified as malware, it also detects what type of malware(trojan-horse, worm, logic-bomb, spyware) it is with a test accuracy of 97.3%. It should be noted that the system takes more time during testing because a dynamic analysis needs to be done on each and every binary.Imad et al. 2019 [27] has used a random forest based classifier trained on a dataset with 211,067 binary samples. It gives a test accuracy of 99.74%.

Table 1 summarizes the works considered above.

| **Reference Number** | **Model/Learning algorithm** | **Dataset Size(number of binary samples)** | **Test Accuracy(in %)** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| [4] | Rule-based classifier using Objective Oriented Association mining | 29,580 | 64.14 |
| [3] | Association mining | 273 | 90 |
| [25]* | DNN | 400,000 | 95 |
| [9]* | ANN | 2,600,000 | 99.8 |
| [10]* | DNN | 50, 000 | 95.64 |
| [11] | Ensemble of Decision Tree, KNNs, Random Forest and SVM | 10,400 | 98.7 |
| [12] | Ensemble of Decision Tree, Random Forest, KNN, Logistic Regression, Linear Discriminant Analysis and Naive Bayes | 5,210 | 98% |
| [13] | CNN | 400,000 | 99.19 |
| [15] | CNN | 70,000 | 95 |
| [26]* | DNN | 10,000 | 97.3 |
| [27] | Random Forest | 211,067 | 99.19 |

The works marked with a * use DNN based classifiers and are the base papers of the work presented in this paper. The dataset and test accuracy of each work should be keenly observed in Table 1. Works with high test accuracy have considered very small datasets with the exception of [9] and works which have considered large datasets have poor accuracy. The work presented in this paper is an attempt to close this gap. A large, industry-level dataset is used to train the system and it gives good test accuracy.

## 3. Deep Learning based Malware Classifier(DLMC)

According to Kaspersky, a popular antimalware company, it scans more than 360,000 new malicious samples every single day. There is an urgent need to come up with an intelligent system which can classify a new, unknown binary classifier (executable/library) as benign or harmful. There are various techniques used to detect malware. Idika et al. 20007 [28] does comprehensive research on different conventional malware detection and analysis techniques. Traditional techniques include Static analysis, Dynamic analysis on the new binary and signature-based detection methods. Each of these traditional methods has drawbacks. Static analysis is the analysis approach done on the binary without running it. Parsing and inspecting the file headers, other data structures present in the file, searching for malicious code sequences in the text section, searching for specific strings, URLs and more without running the file. This can generate a lot of information and that can help in classifying the binary. But it is almost sure that static analysis cannot catch all the details. There can be many true negatives ie., malware being classified as benign which arestill harmful. Andreas et al. 2007 [29] does extensive research on the limitations of static analysis for malware detection. Dynamic analysis is about performing an in-depth analysis on the binary by running it in an isolated environment. What code paths get executed, understanding if any code path behaves in a malicious manner, checking if it tries to download something from an unknown URL/IP Address, does it install harmful services or change system registry keys and more. Dynamic analysis is more effective than static analysis and there are more chances of detecting malware. The main drawback is that dynamic analysis on a binary is time consuming. Coming to signature-based detection methods, it is about finding unique signatures of an already present malware in the database. If the hash of the current malware sample matches with some other malware in the database, it means these two malwares are the same. Not only the complete binary's signature is taken, signatures of peculiar, unique, malicious code sequences are also taken. If they are found in the current malware sample, then it is classified as malicious. The obvious drawback with this method is when a new, unknown binary is to be analyzed, its signature may not be found in the database and the analysis' result is inconclusive - it does not inform us if it is benign or malicious. That was one side of the argument - the method and techniques being used to detect malware are not efficient and ineffective in catching new, unknown malware. Reinforcing this argument is the increasing complexity of modern malware. Modern malwares present themselves in all kinds - python scripts, powershell scripts, bash scripts, executables, libraries, PHP code, JS code and some malwares are fileless in nature. They directly attack a processing already running on the machine thereby reducing its fingerprint in the system. Advanced obfuscation(hiding) techniques, encryption, packing techniques are used to hide malicious code. Ilsun et al. 2010 [30] discusses various advanced obfuscation techniques. Various techniques are used to bypass dynamic analysis. Modern malware can be polymorphic in nature, and it can be compared to the HIV biological virus. It may change its internal code structure regularly, overwrite its own code with something else - everything to make sure it remains undetected by the system. The traditional methods are unable to bear such high sophistication.With the reasons stated above, a radically different approach is needed to attack the problem of malware detection. There is a need for an intelligent malware detection system which can learn its best from the data it has seen so far, and it should be able to catch unknown, new malware which are scanned every day. This work of building an intelligent system was chosen as part of the lab component. Because Windows is the most targeted operating system by the malware authors, this system attempts to detect if a given windows binary(executable/library) is benign or malicious. It is a binary classification problem. As detailed in future subsections, a Deep Neural Network(DNN) based malware detection system is designed, implemented and trained against a large, industry-level dataset

## *3.1. Artificial Neural Networks and Deep Learning*

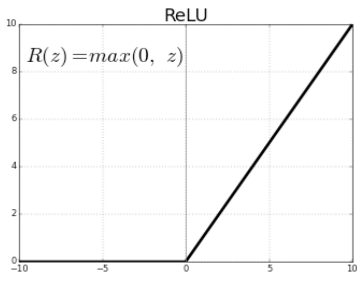
From the above section, it can be observed that, the system should learn from the data available, and it needs to detect new malware. One concept which has proven to do this with high accuracy is Machine Learning (ML). Machine Learning has been used to solve such similar complex problems in various fields.

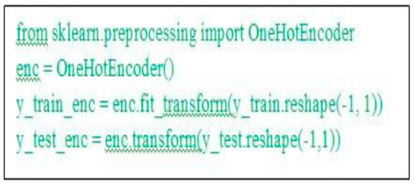
### 3.1.1 Artificial Neural Networks (ANNs)

ANNs is a learning model which is inspired by the biological neural networks present in an animal's brain. It has 3 layers - input layer, hidden layer(s) and an output layer. Each of these layers is an array of an elementary building block known as an artificial neuron. In case of ANN with 2 hidden layers, there are 3 artificial neurons in the input layer, 'n' neurons in each of the hidden layers and 1 neuron in the output layer. The activation function in each neuron which governs that particular neuron's input. This is analogous to the firing of a biological neuron when a certain event(input) happens. A few examples of standard activation functions used are

- Rectified Linear Unit(ReLU), Sigmoid function, Tanh and SoftMax

Out of the above activation functions, the most used functions are ReLU and Softmax. Figure 1 depicts describes ReLU, which is an effective activation with one disadvantage. There is no upper bound on the output value. For which, it is used as an activation function in the hidden layers. For output layer, the output needs to be bounded. That is why the Softmax function is used. Figure 2 describes Software in short.Output of the Softmax function presented in (1) is always between 0 and 1 and any ANN is initialized with random weights(Wi) and





```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
y_train_enc = enc.fit_transform(y_train.reshape(-1, 1))
y_test_enc = enc.transform(y_test.reshape(-1,1))
```

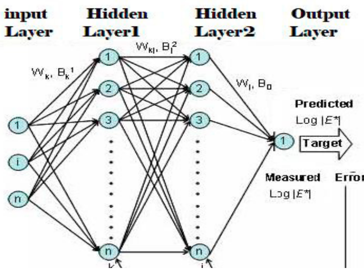


Fig. 1:Softmax of ReLU Fig. 2:Software Softmax Fig. 3:Multi Layered ANN

$$Softmax\ \ f_i(\bar{a}) = \frac{e^{a_i}}{\sum_k e^{a_k}} \ ----(1)$$

On the above functions, the Binary CrossEntropy is an intuitive and effective loss function is calculated as equation (2)

$$H_p(q) \ = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log\big(p(y_i)\big) + (1-y_i) * \log\big(1-p(y_i)\big) \ ----- (2)$$

An algorithm called Back-Propagation is used to reduce the loss function. The model is trained a certain number of times(epochs) and by the end of certain epochs, the model will be ready for testing.

3.1.2 Deep Neural Networks(DNNs)

Deep Neural Networks (DNNs) presented in figure 3 is a variant of ANNs with a large number of hidden layers, higher the number of layers, more complex features are learned by the model. DNNs outperform only when dataset is large, but one issue is also associated with DNNs is over-fitting when training error is very less but testing error is high ie., the network is not performing well on unknown inputs. Note that this is very crucial in our system. The main goal of our system is to detect new, unknown binary samples. There are various regularization techniques to reduce over-fitting. Two effective methods are Batch normalization and dropout and, in our approach, these two methods incorporated to avoid such scenarios. Dropout [30] is a regularization method, where neurons will be dropped out of the network with certain deterministic probability while training the network.

## *3.2. One-Hot Encoding*

When data consists of categorical values other than numerical variables, they need to be encoded into numbers and later fed to the machine learning model as an input. One such standard encoding technique is One-Hot Encoding. Table 2 depicts the colour code sequence where one column contains serial number with respect to one specific colour of second column of the table.

Table 2.Colour Encoding Approach

| Serial No. | Color |
|---|---|
| 1 | Blue |
| 2 | Green |
| 3 | Yellow |
| 4 | Red |
| 5 | Red |
| 6 | Green |

Table 3.Hot Encoding Approach

| Serial No. | Red | Green | Blue | Yellow |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |

As collected data is in form of strings and it cannot be fed into the learning algorithm directly, so we have encoded into an array of numbers and the encoded form isfeed as input. Table 3depicts the one-Hot encoded form of the data present in table 2. one-Hot Encoding is a data pre-processing technique and this one-Hot Encoded data consumes very less memory than corresponding categorical values which typically are character strings.

### *3.3. Portable Executable and Common Object File Format*

Every operating system recognizes a specific executable file format. It can understand only certain executable and library file formats. The Portable Executable(PE)file format is the default format of all the executables and Dynamic Link Libraries(DLLs) generated targeting the Windows operating system [31]. Like any other executable file format(ELF of *NIX, Macho of MAC OS X), it has a file header and the rest of the file is divided into sections. The Common Object File Format(COFF) is the default re-locatable file format used [28]. Both of these file formats have a lot of metadata about the file which can be essential to detect if the binary is benign or malicious.

## **4. Deep Learning based Malware Detection Model**

Works presented in [25], [9], [26], and[10]are taken as the base for work presented in this paper. One persistent problem is seen across these papers described in section 2. Methods which present high test accuracy have been trained on really small datasets and methods which have been trained with large datasets don't perform well. The work presented in this paper attempts to even out this trend. The proposed approach presents an application of DNNs in the field of computer security and the major contributions are highlighted.

1. A DNN-based malware detection system with an aim to detect new, never-seen-before malware.
2. Results and limitations are discussed in details which pave way for more future work.

Although vectorising helps to a certain extent, but the vectorised dataset is still not suitable to train the model completely. A few of its columns have categorical values and it should be noted that machine learning models work on numerical and not on strings. These categorical values need to be converted to a numerical form, so all these columns which are numerical are left alone and other columns' data is encoded with the help of One-Hot encoded method. The following snippet shows sample code for One-Hot encoding.

### *4.1. MLMD Model Approach*

A DNN based learning model is used to build the detection system. The following are the specifications of the model.

1. The model has 9 layers. One input layer, one output layer and 7 hidden layers.
2. The number of neurons in the hidden layers - [2048, 1536, 1024, 512, 256, 128, 32]
    a. ReLU is used as the activation function for all hidden layer neurons.
    b. Two regularization techniques - Batch Normalization and Dropout are used to reduce overfitting. A dropout probability of 0.5 is used on all the hidden layer neurons.
3. Malware detection is a binary classification problem. A given binary sample can be either benign or malicious. For this reason, there are 2 neurons in the output layer.
    a. Softmax is used as the activation function for the output layer neurons.
4. The Binary Cross-Entropy loss function is used to calculate the loss of the model.
5. All the layers are fully connected.
6. BackPropagation is used to modify the parameters.

### *4.2. Data Set*

This paper presents a malware detection system for the windows operating system. For malware detection purpose multiple datasets with PE binary samples were considered. Endgame Malware Benchmark for Research(EMBER)[30], an industry-standard real world dataset with 1.1 million binary samples is used in this paper. It contains 900,000 training samples out of which 300,000 samples are benign, 300,000 malicious and 300,000 unlabeled. Rest of 200,000 samples are used for testing out of which 100,000 are benign and 100,000 are malicious. Every binary sample in the dataset is described by the following features.

1. SHA256 (Secure Hash Algorithm 256) hash of the sample
2. MD5(Message Digest version 5) hash of the sample
3. Year and month first encountered
4. Byte-histogram: Because this is a binary, it can be considered as an array of bytes. A raw byte's value can vary from 0-255. A 256-column histogram can be generated for every sample.
5. Entropy of the sample (using Shannon's entropy formula)
6. PE header details including
    a. File size of the sample in bytes
    b. Size of the sample when in memory in bytes
    c. Timestamp
    d. Target processor architecture(x86, x64, ARM etc.,)
    e. Type of binary(Executable Image / DLL)
    f. Size of code section in bytes
    g. Size of headers
7. List of all the DLLs the binary depends on
8. List of all functions of every DLL the sample depends onand many more features.

## 5. Experiments & Results Discussion

### *5.1. Vectorizing the Dataset*

The EMBER dataset is natively in json format, where every binary sample is described by a rich set of features. Json format increases human readability but is very inefficient (consumes significantly large amounts of memory) and is not a suitable input format for machine learning models. The dataset needs to be vectorized and then fed to the model.

### *5.2. Training and Results*

The model described in section 4.1is trained using vectorized dataset, one-hot encoded dataset containing data about 600,000 windows binaries for 30 epochs. It is tested against 200,000 binary samples. A test accuracy of 96.76% is achieved. Table 4 depicts a comparison of different model's performance with the presented models. In order to improve the system performance an offensive generative model based on GANs is built to make the current DNN-based system more robust. The test accuracy of the improved model is achieved upto 97.42%.

Table 4 Performance Comparison

| Reference Number | Model/Learning algorithm | Dataset Size(number of binary samples) | %Dataset used for training | %Dataset used for testing | Test Accuracy(in %) |
|---|---|---|---|---|---|
| [21] | DNN | 400,000 | 81.04 | 18.96 | 95 |
| [5] | ANN | 2,600,000 | 58% | 42% | 96.8 |
| [6] | DNN | 50, 000 | 90% | 10% | 95.64 |
| [22] | DNN | 10,000 | 75% | 25% | 95.3 |
| DLMD Model | DNN | 600,000 | 75% | 25% | 96.76 |
| GAN-DLMDModel | DNN | 600,000 | 75% | 25% | 97.42 |

Work presented in this paper strikes a balance among 3 parameters like Size of the dataset, amount of dataset used for testing and test accuracy. Although the accuracy of this paper's work is a bit high then the other existing models, but it is consistent and robust against new samples because it has been tested with a good number of samples.

## 6. Conclusion

In this paper, a simple deep neural network-based windows malware detection system is described which achieves a test accuracy of 96.76% which is trained over 0.6 million binary samples. An improved offensive generative model based on GANs is built to make the current DNN-based system more robust with test accuracy of 97.42%.This work proves that Deep Neural Networks combined with rigorous static analysis can help building an intelligent malware detection system. The advantage of using DNNs is that it can learn complex features with a greater number of layers and more data.

## References

[1]. Kaspersky Labs, "Kaspersky Lab detects 360,000 new malicious files daily – up 11.5% from 2016, " [Online] Available: https://www.kaspersky.com/about/press-releases/2017_kaspersky-lab-detects-360000-new-malicious-files-daily [Accessed: May. 3, 2020]

[2]. Anderson, H. S., & Roth, P. (2018). Ember: an open dataset for training static pe malware machine learning models. arXiv preprint arXiv:1804.04637.

[3]. Ravi, C., & Manoharan, R. (2012). Malware detection using windows api sequence and machine learning. International Journal of Computer Applications, 43(17), 12-16.

[4]. Filho, A. S., Rodríguez, R. J., & Feitosa, E. L. (2022). Evasion and Countermeasures Techniques to Detect Dynamic Binary Instrumentation Frameworks. Digital Threats: Research and Practice (DTRAP), 3(2), 1-28.

[5]. Lad, S. S., & Adamuthe, A. C. (2022). Improved Deep Learning Model for Static PE Files Malware Detection and Classification. International Journal of Computer Network & Information Security, 14(2).

[6]. Mane, T., Nimase, P., Parihar, P., & Chandankhede, P. (2022). Review of malware detection using deep learning. In Soft Computing for Security Applications (pp. 255-262). Springer, Singapore.

[7]. Li, C., Lv, Q., Li, N., Wang, Y., Sun, D., & Qiao, Y. (2022). A novel deep framework for dynamic malware detection based on API sequence intrinsic features. Computers & Security, 116, 102686.

[8]. Anil Kumar, D., Das, S. K., & Sahoo, M. K. (2022). Malware Detection System Using API-Decision Tree. In Advances in Data Science and Management (pp. 511-517). Springer, Singapore.

[9]. G. E. Dahl, J. W. Stokes, L. Deng and D. Yu, "Large-scale malware classification using random projections and neural networks," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, 2013, pp. 3422-3426* [Accessed: May. 3, 2020]

[10]. W. Hardy, "DL4MD: A deep learning framework for intelligent malware detection," *Proceedings of the International Conference on Data Mining (DMIN) 2016.* [Accessed: May. 3, 2020]

[11]. Vyas, Rushabh& Luo, Xiao & McFarland, Nichole & Justice, Connie, "Investigation of malicious portable executable file detection on the network using supervised learning techniques" *941-946. 10.23919/INM.2017.7987416* [Accessed: May. 3, 2020]

[12]. Kumar, Ajit&Kuppusamy, K S &Gnanasekaran, Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *Journal of King Saud University - Computer and Information Sciences. 31. 10.1016/j.jksuci.2017.01.003* [Accessed: May. 3, 2020]

[13]. J. Johns, "Representation Learning for Malware Classification - 2017," [Online] Available: https://www.fireeye.com/content/dam/fireeye-www/blog/pdfs/malware-classification-slides.pdf [Accessed: May. 3, 2020]

[14]. M. D.Khan, M. T. Shaikh, R. Ansari, M. Suriya, S. Suryawanshi, "Malware detection using Machine Learning Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering,* ISO 3297:2007 Certified, Vol. 6, Issue 9, September 2017 [Accessed: May. 3, 2020]

[15]. Z. Kan, H. Wang, G. Xu, Y. Guo, X. Chen, "Towards Light-weight Deep Learning based Malware Detection," *2018 42nd IEEE International Conference on Computer Software & Applications* [Accessed: May. 3, 2020]

[16]. Lu, Le, et al. "Deep learning and convolutional neural networks for medical image computing." *Advances in Computer Vision and Pattern Recognition* (2017) [Accessed: May. 3, 2020]

[17]. A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, *542*(7639), 115-118. [Accessed: May. 3, 2020]

[18]. Choudhury, S., Karthik Pai, B. H., Hemant Kumar Reddy, K., & Roy, D. S. (2022). A Hybrid CNN Real-Time Object Identification and Classification Approach for Autonomous Vehicles. In Intelligent Systems (pp. 485-497). Springer, Singapore.

[19]. Wu, Tingmin, et al. "Twitter spam detection based on deep learning." *Proceedings of the australasian computer science week multiconference*. 2017 [Accessed: May. 3, 2020]

[20]. Le, Hung, et al. "URLnet: Learning a URL representation with deep learning for malicious URL detection." *arXiv preprint arXiv:1802.03162* (2018). [Accessed: May. 3, 2020]

[21]. Javaid, Ahmad, et al. "A deep learning approach for network intrusion detection system," *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. 2016.[Accessed: May. 3, 2020]

[22]. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (*NIPS'14*). MIT Press, Cambridge, MA, USA, 2672–2680. [Accessed: May. 3, 2020]

[23]. Hu, Weiwei, and Ying Tan, "Generating adversarial malware examples for black-box attacks based on gan," *arXiv preprint arXiv:1702.05983* (2017). [Accessed: May. 3, 2020]

[24]. Lin, Zilong, Y. Shi and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," *arXiv preprint arXiv:1809.02077* (2018). [Accessed: May. 3, 2020]

[25]. Saxe, Joshua, and Konstantin Berlin, "Deep neural network based malware detection using two dimensional binary program features," *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)* [Accessed: May. 3, 2020]

[26]. W. Yin, H. Zhou, M. Wang, Z. Jin, J. Xu, "Dynamic Malware Detection Mechanism Based on Deep Learning," *IJCSNS International Journal of Computer Science and Network Security*, VOL.18 No.7, July 2018 [Accessed: May, 3. 2020]

[27]. I. Abdessadki and S. Lazaar, "A New Classification Based Model for Malicious PE Files Detection," *International Journal of Computer Network and Information Security*. 1-9. 10.5815/ijcnis.2019.06.01 [Accessed: May. 3, 2020]

[28]. N. Idika and Aditya P. Mathur, "A survey of malware detection techniques," *Purdue University* 48 (2007): 2007-2 [Accessed: May. 3, 2020]

[29]. A. Moser, C. Kruegel and E. Kirda, "Limits of Static Analysis for Malware Detection," *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, FL, 2007, pp. 421-430 [Accessed: May. 3, 2020]

[30]. I. You and K. Yim, "Malware Obfuscation Techniques: A Brief Survey," *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, Fukuoka, 2010, pp. 297-300 [Accessed: May. 3, 2020]

[31]. Microsoft, "PE Format - Win32 apps," [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format [Accessed: May. 3, 2020]