digital

MDPI

Article

Web-Based Malware Detection System Using Convolutional Neural Network

Ali Alqahtani [1,2,*], Sumayya Azzony [1], Leen Alsharafi [1,†] and Maha Alaseri [1,†]

[1] Department of Computer Science, King Khalid University, Abha 61421, Saudi Arabia; sazzony@kku.edu.sa (S.A.); leenalsharafi0@gmail.com (L.A.); alaserimaha@gmail.com (M.A.)
[2] Center for Artificial Intelligence (CAI), King Khalid University, Abha 61421, Saudi Arabia
* Correspondence: amosfer@kku.edu.sa
† These authors contributed equally to this work.

Abstract: In this article, we introduce a web-based malware detection system that leverages a deep-learning approach. Our primary objective is the development of a robust deep-learning model designed for classifying malware in executable files. In contrast to conventional malware detection systems, our approach relies on static detection techniques to unveil the true nature of files as either malicious or benign. Our method makes use of a one-dimensional convolutional neural network 1D-CNN due to the nature of the portable executable file. Significantly, static analysis aligns perfectly with our objectives, allowing us to uncover static features within the portable executable header. This choice holds particular significance given the potential risks associated with dynamic detection, often necessitating the setup of controlled environments, such as virtual machines, to mitigate dangers. Moreover, we seamlessly integrate this effective deep-learning method into a web-based system, rendering it accessible and user-friendly via a web interface. Empirical evidence showcases the efficiency of our proposed methods, as demonstrated in extensive comparisons with state-of-the-art models across three diverse datasets. Our results undeniably affirm the superiority of our approach, delivering a practical, dependable, and rapid mechanism for identifying malware within executable files.

Keywords: malware; cyber security; deep learning; machine learning; deep neural networks

Citation: Alqahtani, A.; Azzony, S.; Alsharafi, L.; Alaseri, M. Web-Based Malware Detection System Using Convolutional Neural Network. *Digital* **2023**, *3*, 273–285. https://doi.org/10.3390/digital3030017

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 25 July 2023
Revised: 7 September 2023
Accepted: 8 September 2023
Published: 12 September 2023

1. Introduction

Although the internet has become essential, its security is seriously threatened by malicious software (malware), which refers to any parasitical and destructive software developed by hacking data, damaging or destroying computer systems. This malware continues escalating and poses a significant security concern in the digital age. Exponential malware attacks have influenced many computer users, corporations, and governments, making malware detection an open research topic. Some current malware detection approaches adopt conventional, time-consuming methods. They have proven ineffective in identifying unknown malware in real time because of a distinct digital trace, a standard signature-based method. Programs are inspected with antivirus applications, evaluating their signatures against known malware. Most existing malware applications tend to avoid intelligent detection, such as code encryption, obfuscation, and polymorphism, or by changing their signature, making it easier for evils to modify malware code.

Deep learning has been used more frequently in recent years, thereby becoming a major topic in variety of fields. It has extended into broader and deeper models, especially in detection and classification. The deep-learning methods are introduced to detach the data manifolds and allow a classification method to deal with learned representation rather than raw data [1]. Conventional classification algorithms achieve limited performance as dimensionality increases. Therefore, dealing with high-level features supports the achievement of classification tasks. This paper developed an approach to detect and

alarm users whether the portable executable file (PE file), which commonly has a .exe file extension and contains instructions that the system executes immediately when the user clicks on those files. The executable files are software but are exploited by many hackers to create harmful programs that can damage the computer system. The introduced method has considered the power of deep learning when classifying such files. We target the header part of PE files, mainly the PE header, as it contains essential metadata with all the necessary information the operating system requires to execute such files. The proposed approach aims to detect malware new to the operating systems' anti-viruses and make the detection process more accessible and effective without any manual process. The comprehensive pipeline of our introduced deep detection of malware is presented in Figure 1. The proposed method uses 1D-CNN due to the nature of the portable executable file. The 1D-CNN is practical and recommended when working with a specific datatype [2]. The effectiveness of the proposed methods is demonstrated on several datasets compared to the most recent methods. The finding results show that the proposed approach efficiently detects malware files with comparable accuracy. Our web-based malware detection system has far-reaching applications in the cybersecurity landscape, benefiting individuals and organizations alike. It offers a user-friendly solution for individuals seeking to safeguard their devices against malware threats, empowering them to assess file integrity before execution. This reduces susceptibility to cyberattacks and data breaches, even for users with limited technical expertise. Within corporate environments, our system functions as an active defense mechanism. Rapidly identifying possible malicious files strengthens incident response, limits malware propagation, and safeguards sensitive data.

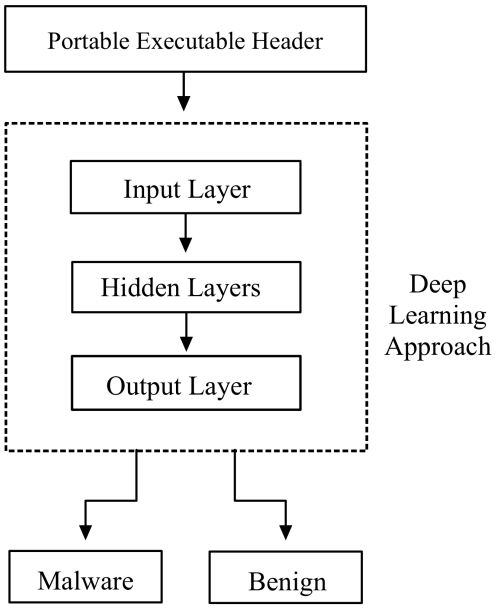

Figure 1. General structure of the malware detection model [3].

The rest of this paper is organized as follows. In Section 2, we present related works, while we describe our proposed methodology in Section 3. In Section 4, we present our experimental results, and the deployment step is presented in Section 5. Finally, concluding remarks are provided in Section 6.

2. Related Work

Machine learning algorithms have continuously demonstrated high discriminatory abilities across a wide range of malware detection systems in earlier research. In recent times, deep learning has also been used in malware detection systems, among other domains. Making the trained model accessible and usable via a web interface is necessary when deploying a deep-learning model in a web-based application. Utilizing a deep-

learning methodology, we created a web-based malware detection system. Our website is the first of its kind in terms of malware detection methods, as far as we know. We employed deep-learning modeling and deep-learning deployment to classify the earlier systems as you can see in the next two subsections.

2.1. Deep-Learning Modeling

Malware detection has received considerable attention. Most existing methods vary from standard signature-based detection [4–6] to more recent machine learning (ML) [7–9] and deep-learning (DL) [10–12] detection. Conventional malware detection based on signature has been ineffective against a wide range of variations since all the current malware applications tend to avoid detection. In addition, all modern-day malware comes with intelligent approaches such as code encryption, obfuscation, and polymorphism. Hence, the most recent and active research has advanced to more advanced methods, such as deep learning and machine learning. In ML approaches, typically supervised learning algorithms are applied to the collected features of files for detection (whether the file is benign or malicious) or even for classification (by categorizing whether it is malicious under a particular class of malware).

Several supervised machine learning algorithms have been proposed for malware detection. Sharma et al. [13] introduced a framework for detecting unknown malware using machine learning techniques based on the frequency of opcode occurrence. Their work applied Random Forest, J48 Graft, Logistic Model Tree (LMT), and Naïve Bayes Tree (NBT). Hussain et al. [14] also analyzed the features of the portable executable (PE) header and excluded others that do not help the model's performance. The prediction layer was the focus of various ML techniques [14], including Random Forest (RF), Decision Tree, AdaBoost, Support Vector Machine, Gradient Boosting(GB), and Gaussian Naive Bayes (GNB). Although these methods proved successful in the detection task, the results show that RF performs much better. Kolter, J. Z. [15] applied n-grams of byte codes to benign and malicious executable files in their dataset. They explored Naive Bayes, Decision Trees, Support Vector Machines, and boosting algorithms, among other inductive methods. Schultz et al. [16] also presented a data-mining framework that accurately and automatically detects new, previously unseen malicious executable files using the Naive Bayes classifier. Due to some drawbacks of machine learning methods, an effort has been made in the feature extraction phase. As the PE file consists of numerous features, analyzing which feature to select and which to drop is required. Deep-learning-based techniques are considered to overcome the weaknesses associated with feature selection. This is an effective solution to solve this problem as it does not require the hand-crafted feature, shifting the emphasis to feature extraction and representation learning, which is why deep learning gains a lot of traction in malware detection.

Azeez et al. [17] proposed a method using a hybrid ensemble learning approach comprising fully connected and Convolutional Neural Networks with extra tree classifiers acting as meta learners to detect malware. Rathore et al. [18] also adopted deep-learning methods to build the classification models using deep neural networks (DNN) (DNN-2L, DNN-4L, DNN-7L). In addition, Sewak et al. [19] used one-layer and three-layer stacked auto-encoder (SAE) for feature extraction and dimensionality reduction. Then, the deep neural network with 2-4-7 hidden layers is utilized to classify malicious and benign executable files. Hardy et al. [20] collected Windows API calls from an authentic and extensive file collection. They utilized a deep network, mainly the SAE model, to detect malware. However, dealing with a filter that suits PE files requires a careful approach. Therefore, the 1D-CNN is practical and recommended when dealing with this specific datatype.

The authors of [21] suggest an attack that combines binary diversification methods with frameworks for optimization in order to deceive such DNNs while maintaining the functionality of binaries. They tested their attack in both white-box and black-box environments on three DNNs and discovered that it frequently had success rates. They looked at several defenses, both new and ancient, and discovered ones that could successfully thwart

more than 80% of our evasion attempts. They recommended adding techniques that do not rely on machine learning to malware detection systems because these protections may still be vulnerable to evasion by assaults.

2.2. Deep-Learning Deployment

Deploying a deep-learning model in a web-based application involves making the trained model available and accessible via a web interface. This allows users to interact with the model and obtain predictions or perform various tasks using the model's capabilities. Web-based applications are valuable across various domains, and their usage has been on the rise, particularly in applications incorporating deep-learning techniques. The integration of deep learning within web-based systems has opened up new possibilities and yielded significant advancements in several areas. Here, we highlight a selection of applications that have successfully utilized deep-learning techniques within web-based systems.

Coronel et al. [22] tackle the COVID-19 health emergency by introducing a web application that allows personnel access to a workspace to be controlled without physical touch. Face detection and recognition have been implemented using deep-learning and computer vision techniques. Singla et al. [23] also proposed a smart disaster management system based on deep-learning methods for online decision making. The proposed system helps disaster professionals identify the types of social media posts relevant to disasters. This study organizes the tweets into need-based, availability-based, situation-based, general, and irrelevant categories before visualizing them on a map, according to their geographic location. In addition, Ismail et al. [24] presented a deep-learning facial recognition web-based attendance system. The cutting-edge deep-learning face recognition pre-trained models were coupled with the web environment and connected to an online database for feature storage in the proposed attendance system.

3. Method

The proposed method makes use of a one-dimensional convolutional neural network (1D-CNN), chosen due to its inherent compatibility with the structural characteristics of portable executable (PE) files. The 1D-CNN is practical and recommended when dealing with the PE dataset [2]. 1D-CNN also privileges from CNN characteristics [25]. Learning local features and sharing parameters differentiate CNN to have a property when translating hidden features, showing its high capability of achieving challenging classification tasks. Malware detection has been identified as a binary classification task, where benign is one class and malware is another. This section outlines the fundamental aspects of the 1D-CNN architecture and delves into the intricacies of the proposed model variations.

3.1. One-Dimensional Convolutional Neural Network (1D-CNN)

In contrast to traditional Neural Networks (NNs), Convolutional Neural Networks (CNNs) exhibit a crucial distinction: the integration of convolutional layers that effectively capture spatial features. These convolutional layers utilize the concept of convolution, where localized filters traverse distinct regions of the input space. This process facilitates the acquisition of localized features, serving as a vital mechanism in comprehending complex data patterns. The CNN architecture, known for its multi-layer structure, encompasses distinct phases: the convolution phase, the activation phase, and the pooling phase. These phases collaborate to generate successive feature maps evolving as data traverse the network. Particularly relevant to our application, the 1D-convolutional layer handles feature extraction. By strategically calculating the outputs of the neurons linked to localized input regions, using dot products with a learnable 1D filter, this layer effectively captures pertinent information. Subsequent fully connected layers subject the resulting vector to linear transformations, subsequently modulated using non-linear activation functions to introduce essential non-linearity crucial for accurate predictions

The proposed architecture of the 1D-CNN model for malware detection is illustrated in Figure 2. This diagram showcases how the filter moves within the 1D-CNN mode,

operating in one dimension along the time axis. Each filter corresponds to a series of certain axes in the portable executable files.

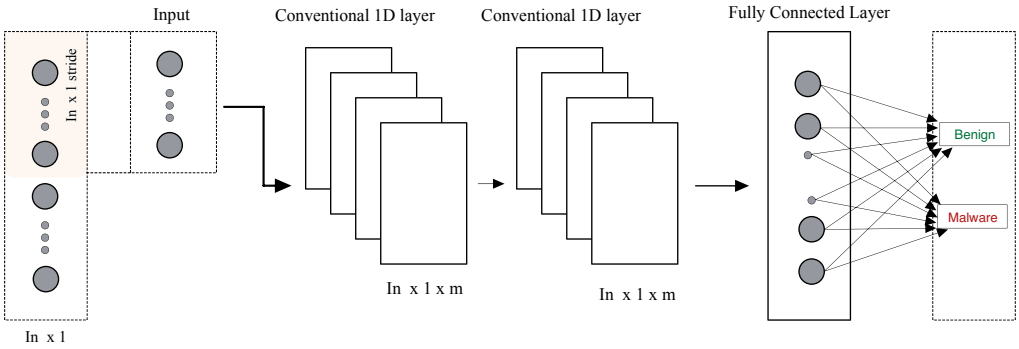

Figure 2. One-Dimensional Convolutional Neural Network (CNN) architecture. Each filter corresponds to a series of certain axes in the portable executable files.

In Figure 2, m represents the number of filters, and 'In' is the input of features. The network comprises two layers: a one-dimensional (1D) convolutional layer and a fully connected one. In the initial layer, the kernel convolves over the input data in one dimension, extracting features. The subsequent layer is used for classification prediction. To enhance our approach, we will further explore the potential of the 1D-convolutional layer in improving feature extraction for our malware detection model.

3.2. Architectures

As shown in Table 1, we proposed four different architectures. The details are provided below.

- MODEL1: 1D-CNN model with four hidden layers, three of them are 1D-convolutional layers containing (16, 32, and 64 filters) with a size of 3, which is followed by one fully connected layer with 128 nodes.
- MODEL2: 1D-CNN model with four hidden layers, three of them are 1D-convolutional layers containing (16, 16, and 16 filters) with a size of 3, which is followed by one fully-connected layer with 128 nodes.
- MODEL3: 1D-CNN model with four hidden layers, three of them are 1D-convolutional layers containing (32, 32, and 32 filters) with a size of 3, which is followed by one fully connected layer with 128 nodes.
- MODEL4: 1D-CNN model with four hidden layers, three of them are 1D-convolutional layers containing (64, 64, and 64 filters) with a size of 3, which is followed by one fully connected layer with 128 nodes.

Table 1. Detailed configuration of the 1D-CNN architectures.

MODEL1	
Layer	Architecture
Convolutional	3 × 1 × 16
Convolutional	3 × 1 × 32
Convolutional	3 × 1 × 64
Fully-connected	128
MODEL2	
Layer	Architecture
Convolutional	3 × 1 × 16
Convolutional	3 × 1 × 16
Convolutional	3 × 1 × 16
Fully-connected	128

Table 1. *Cont.*

MODEL3	
Layer	Architecture
Convolutional	3 × 1 × 32
Convolutional	3 × 1 × 32
Convolutional	3 × 1 × 32
Fully-connected	128
MODEL4	
Layer	Architecture
Convolutional	3 × 1 × 64
Convolutional	3 × 1 × 64
Convolutional	3 × 1 × 64
Fully-connected	128

4. Experiments and Discussion

In this section, the presented results of comprehensive experimentation and subsequent discussions unfold. The introduced datasets underwent normalization for uniformity. Following this, a range of established techniques are explored to establish a baseline for comparison. The chosen evaluation metric is elaborated upon, capturing the essence of the model assessment. The technical implementation details, encompassing tools, and dataset specifics, are outlined. Finally, the core experimental outcomes are presented, highlighting the accuracy of the developed models. This thorough analysis lays the foundation for further discussions and the conclusions drawn from this study's findings.

4.1. Datasets

Our developed model was implemented using different PE header datasets with malware and benign files and was collected from different sources. These datasets were downloaded from an open-source repository:

- Classification of Malware (CLaMP) dataset [26], which was obtained from Mendeley data source, containing 5210 file samples, where 2722 are malware, and 2488 are benign files, with 69 features.
- Benign and Malicious PE Files dataset [27] from the Kaggle community platform with 19,612 samples, where 14,600 are malware, and 5010 are benign, with 79 features.
- MalwareDataSet dataset [28] from the GitHub platform contains a total of 137,444 samples: 40,918 of them are malware, and 96,526 are benign, with nine features.

To make the data on the same scale, data normalization was applied to the datasets by converting the numerical values in the dataset to a standard form using Standard Deviation Normalization using the following formula:

$$xnormalized = (x - \mu)/\sigma \tag{1}$$

where μ is the mean, and σ is the standard deviation. The standard deviation can be calculated using the following formula:

$$\sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \overline{x})^2}{n-1}} \tag{2}$$

where n denotes the number of data samples , x_i represents each of the values of the data, and $\overline{x}$ represents the mean of x_i. All datasets were split into 80% of the total dataset for training and 20% for testing.

4.2. Baseline Methods

In this section, an exploration of the baseline methods employed to establish a reference point for the analysis takes place. The methods fall into two distinct categories: machine learning methods and deep-learning methods.

4.2.1. Machine Learning Methods

The initial category of baseline methods centers on machine learning techniques. The starting point involves the Decision Tree (DT), a rooted tree structure wherein internal nodes signify attributes and leaf nodes correspond to class labels. The traversal of this tree leads to the prediction of the class label of a leaf node, thereby allowing the visualization of relationships amidst features. DT offer the benefit of intuitive visualization, enabling a clear understanding of how different attributes contribute to the final classification. Subsequently, an investigation into the Support Vector Machine (SVM) follows, i.e., an algorithm that segregates data into multidimensional spaces via hyperplanes for the classification of diverse categories. The SVM algorithm strives to maximize the margin between classes to attain optimal classification outcomes. SVM presents the distinct benefit of maximizing the margin between different classes. This enhances its ability to achieve optimal classification results even in complex, multidimensional spaces by effectively identifying the most informative data points, known as support vectors, to make accurate decisions at the class boundaries.

4.2.2. Deep-Learning Methods

The second grouping of baseline methods delves into the realm of deep-learning techniques. The approach undertaken adopts the foundational architecture introduced in [18], which encompasses three deep neural network (DNN) models. These models are categorized as follows:

- Two hidden layers contain 1024 and 32 nodes, named 2L-DNN.
- Four hidden layers contain $2^{12-(2*i)}$ nodes in the ith hidden layer (1024, 256, 64, and 16 nodes), named 4L-DNN.
- Seven hidden layers contain 2^{11-i} nodes (1024, 512, ..., 16 nodes) in the ith hidden layer, named 7L-DNN.

Additionally, exploration extends to the 1D-CNN with LSTM, a neural network architecture harmonizing Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. This amalgamation distinctly caters to recognizing long-term dependencies, particularly advantageous for sequence prediction challenges. The experimental approach incorporates the utilization of 1D-CNN architectures, complemented by the integration of LSTM layers at the culmination of each model's architecture.

4.3. Model Evaluation and Implementation

To assess the quality of the detection model, we computed the accuracy (ACY) evaluation metric on the test data, which is considered the most appropriate based on the data used. The accuracy represents the proportion of correctly classified instances and is calculated using the formula:

$$Accuracy = \frac{Correct\ predictions\ number}{Total\ prediction\ number} \tag{3}$$

For implementation, the proposed methods were implemented using Keras in Python and tested on three distinct datasets: Classification of Malware (CLaMP) [26], Benign and Malicious PE Files [27], and MalwareDataSet [28]. These datasets have been frequently used in prior research on malware detection. The proposed models were trained without pre-training or fine-tuning steps, employing the Adam optimizer with batches of 64 shuffled samples. Model convergence was achieved after 120 epochs, utilizing a learning rate of 0.006, momentum of 0.9, and weight decay of 0.0005. The Exponential Linear Unit

(ELU) was used as the activation function for all network layers. As we require binary classification, the output layer employs the sigmoid activation function. To mitigate overfitting, a dropout layer with a rate of 0.1 was incorporated after each layer.

4.4. Experimental Results

The experimental models are trained to learn feature representation and classify malware in the executable file. Three different Deep Neural Network models are shown in Table 2 and have been evaluated. Figure 3 summarizes the experimental results on the three different datasets. All experimental models have shown comparative results, which allowed us to obtain the baseline accuracies for our study. The 4L-DNN model outperforms other DNN architecture by a significant margin on the accuracy metric, where 98.85% was achieved on both datasets, Benign and Malicious PE Files dataset and MalwareDataSet, and 98.37% was achieved on the Classification of Malwares dataset.

Table 2. Detailed configuration of the fully-connected architectures.

2L-DNN	
Layer	Nodes
Fully-connected 1	1024
Fully-connected 2	32
4L-DNN	
Layer	Nodes
Fully-connected 1	1024
Fully-connected 2	256
Fully-connected 3	64
Fully-connected 4	16
7L-DNN	
Layer	Nodes
Fully-connected 1	1024
Fully-connected 2	512
Fully-connected 3	256
Fully-connected 4	128
Fully-connected 5	64
Fully-connected 6	32
Fully-connected 7	16

The accuracy (ACY) metric was computed for four different 1D-CNN architectures to evaluate the classification quality of the developed methods. The comparative results on the three different datasets are demonstrated in Figure 4. It is noteworthy that due to the nature of the portable executable file, the results varied based on the number of learnable filters, whereas a network with fewer filters reforms much better. We believe the reason behind that is due to the number of input features of the experimental dataset.

To investigate the abilities of our DL methods, we demonstrate their effectiveness against three baselines, including DT, SVM, and DNN. Table 3 demonstrates the comparative results. DL models achieve higher classification performance when compared to ML models. One reasonable explanation is that due to a large amount of input features of the Malwares dataset and Benign and Malicious PE Files dataset, the classification performance of DL methods substantially outperforms the performance of ML methods. It is also worth noting that because the MalwareDataSet dataset has nine input features, ML algorithms perform much better compared to others.

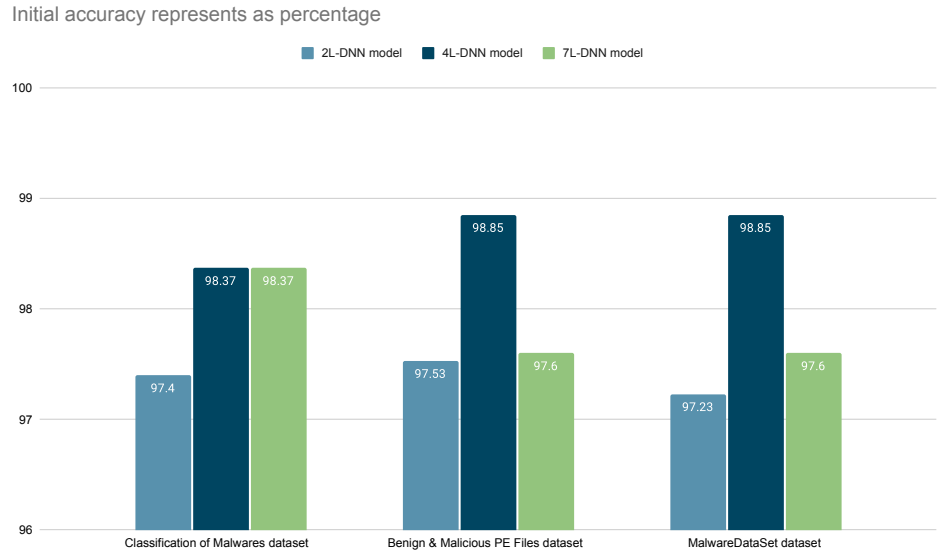

Figure 3. Accuracy chart for datasets with (DNNs) models.

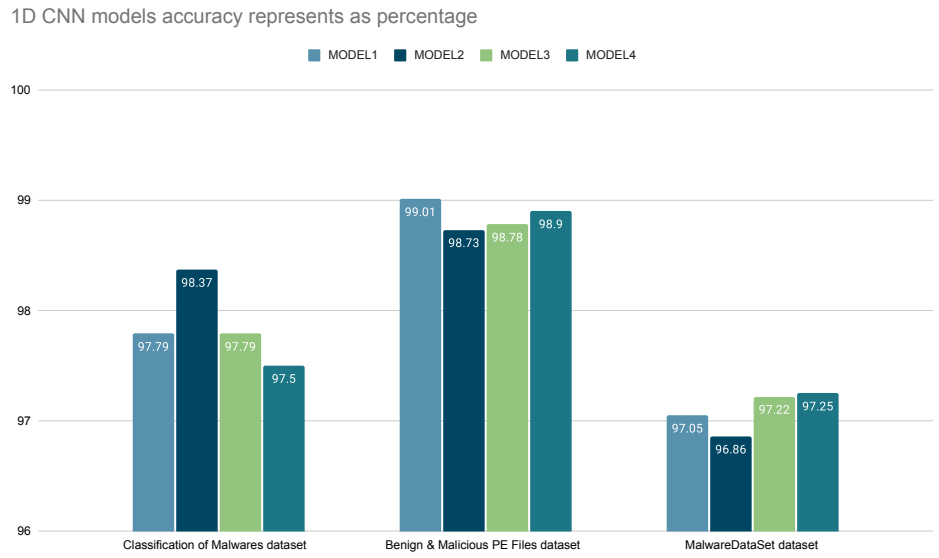

Figure 4. Accuracy chart for datasets with (1D-CNN) models.

Table 3. Accuracy values for datasets with ML and 1D-CNN-LSTM techniques.

Datasets	SVM	DT	DNN	1D-CNN	1D-CNN-LSTM
Classification of Malwares	95.11	96.45	98.37	98.37	98.75
Benign and Malicious PE Files	96.13	98.65	98.85	99.01	98.80
MalwareDataSet	91.67	98.80	98.85	97.25	97.07

5. Web-Based System

The selection of a web-based framework for our malware detection system is underpinned by the desire to provide enhanced accessibility and usability to Windows users, addressing the specific cybersecurity challenges faced by this user base. Our deployment of a deep-learning model within a web interface seeks to bridge the gap between sophisticated malware detection technology and the specific requirements of Windows users, thereby catering to the evolving landscape of cyber threats. By embracing a web-based interface, our approach offers several distinct advantages tailored to the needs of Windows users. Foremost among these is the intuitive and user-friendly experience provided, a notable

departure from traditional malware detection methods that often involve intricate setup processes and specialized knowledge. This design choice ensures that even users with limited technical expertise can benefit from our malware detection capabilities without requiring extensive training or prior familiarity with cybersecurity practices. We chose MODEL1 to deploy our malware detection web-based system due to its superior accuracy results. The model is based on a 1D convolutional neural network (1D CNN) architecture that takes as input a sequence of features extracted from a PE file and predicts whether it is malicious or benign. The first step, before deploying the trained MODEL1 to our malware detection web-based system, was to save the model in a proper format using the save method. We saved the trained model into h5 format, which is a commonly used format for saving and loading deep-learning models. Then, we started to develop the front end, which involves creating a web-based interface that allows users to upload files and view the results of the malware detection process. The front end of the system was developed using HTML, CSS, and JavaScript. The design of the user interface included a single web page with five main sections: Home, Upload, About the Project, Team Members, and Contact Us Sections shown in Figure 5. The Upload Section is the main focus of the webpage, providing users with a clear and simple interface to upload files. This section includes a drag-and-drop area to drop files for upload with the addition of error handling mechanisms, which include techniques such as validation checks by ensuring the input file meets the required format before it is processed via the model. The back-end system for the malware detection tool was implemented using Flask framework [29], a lightweight web framework for Python that allows us to create a simple web page where the user can upload a PE file and obtain an immediate classification result. With the help of the pefile library (Note: https://github.com/erocarrera/pefile, accessed on 25 July 2023) that is used to extract the header features of the uploaded PE files, including its size, entry point, and imported functions. These features are then used to create a feature vector that will be passed through several processing steps where it must be normalized to ensure that all attributes have the same scale, and reshaped into 1D form. This is because the 1D CNN is designed to operate on sequences of data. Finally, the model makes a prediction on the reshaped feature vector. The prediction is a binary classification, where the two classes correspond to benign and malware. The output of the model is a probability distribution. To obtain a single prediction, the maximum function was used, which returns the index of the class with the highest probability. This index is then used to determine the status of the file. The results of the model prediction are flashed to the user interface. The code, data, and implementation of our web-based system are available at: https://github.com/LeenAlsharafi/MalwareDetectionSystem.git (accessed on 25 July 2023).

Comparison with Existing Web-Based Systems

Based on our literature review, there is no paper that has introduced a convolutional neural network-based web-based malware detection system. So, we have compared our malware detection web-based system with the other web-based systems mentioned in the related work generally. Using an open-source, pre-trained deep-learning model, this study [22] suggests a web-based attendance system that uses facial recognition. In contrast, our proposed web-based malware detection system was trained with different machine learning models and deep-learning models. Only bidirectional long short-term memory network layers and a convolution neural network are used in the suggested model [23], which introduced a web-based smart disaster management system that can analyze social media messages to help in decision making. This web-based system and the one we suggest are the first of their kind in the field. In paper [24], several experiments have been carried out to evaluate the proposed system relying on training deep-learning models only unlike our proposed system, which trained different machine learning models and deep-learning models for evaluation purposes.

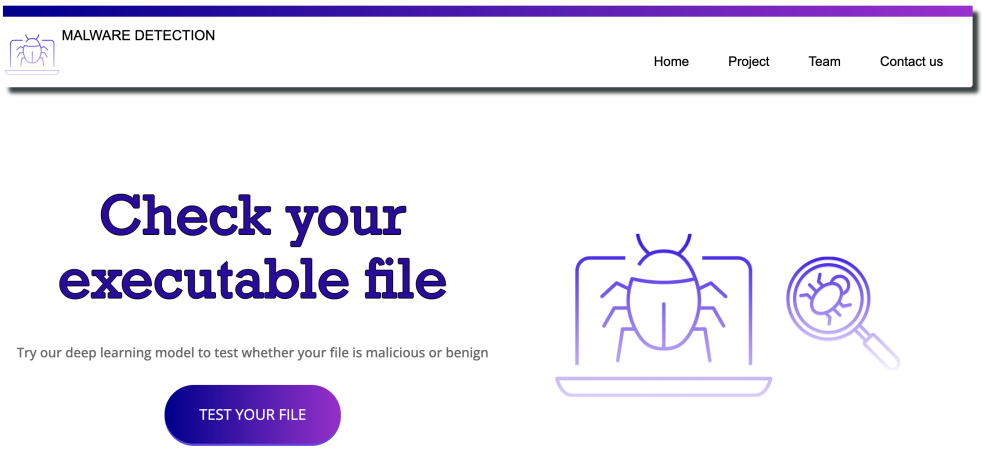

Figure 5. The design of the user interface is simple and intuitive, with clear visual cues and minimal distractions. The Home section of the webpage provides a brief statement as an introduction to the system. The Upload section offers users a clear and simple interface to upload files. The Project section presents more detailed information about the system. The Team Members section lists the individuals who developed the system. Finally, the Contact Us section features a straightforward form that users can complete to send a message to the system's support team.

6. Conclusions

In conclusion, this paper has presented a novel and effective deep-learning approach for the classification of malicious software based on portable executable files. The utilization of a one-dimensional convolutional neural network (1D-CNN) is particularly suitable for analyzing the header portions of these files, contributing to the development of an efficient web-based malware detection system. While the results showcased the effectiveness of our proposed method, it is essential to acknowledge certain limitations that warrant consideration. First and foremost, the ever-increasing sophistication of malware presents an ongoing challenge. Malicious actors continue to devise new techniques, obfuscation methods, and evasion strategies, thereby rendering the task of malware detection more intricate. Additionally, the reliance on header-based features, although effective, may not capture the full spectrum of features that distinguish advanced malware from benign files. Consequently, our current approach might encounter difficulties in identifying more subtle and polymorphic forms of malware. As a direction for future research and extension of this work, we intend to address these limitations by incorporating a broader and more diverse range of malicious data into the dataset. This augmentation would enable our model to learn from an increasingly representative set of malware samples, thereby enhancing its ability to detect a wider array of malicious instances, including those belonging to specific categories such as ransomware. Moreover, we envision the evolution of our detection system to not only identify the presence of malware but also classify the detected malware into distinct types or families. By incorporating more sophisticated classification algorithms, such as ensemble methods or deep-learning architectures designed for multi-class classification, we aim to provide deeper insights into the nature of the detected malware. This extension would empower security professionals and system administrators with more granular information about the threats they face, thereby facilitating more targeted and effective countermeasures.

Author Contributions: Conceptualization, A.A., S.A., L.A. and M.A.; methodology, A.A., S.A., L.A. and M.A.; software, L.A. and M.A.; validation, L.A., M.A. and S.A.; formal analysis, L.A. and M.A.; investigation, A.A., S.A., L.A. and M.A.; resources, A.A., S.A., L.A. and M.A.; data curation, A.A., S.A., L.A. and M.A.; writing—original draft preparation, L.A. and M.A.; writing—review and editing, A.A., S.A., L.A. and M.A.; supervision, A.A. and S.A.; project administration, A.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Deanship of Scientific Research, King Khalid University, Saudi Arabia, under Grant number (RGP2/332/44).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alqahtani, A.; Xie, X.; Deng, J.; Jones, M.W. A deep convolutional auto-encoder with embedded clustering. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 4058–4062.
2. Alqahtani, A.; Ali, M.; Xie, X.; Jones, M.W. Deep time-series clustering: A review. *Electronics* **2021**, *10*, 3001. [CrossRef]
3. Aslan, Ö.A.; Samet, R. A comprehensive review on malware detection approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
4. Malhotra, A.; Bajaj, K. A hybrid pattern based text mining approach for malware detection using DBScan. *CSI Trans. ICT* **2016**, *4*, 141–149. [CrossRef]
5. Baldangombo, U.; Jambaljav, N.; Horng, S.J. A static malware detection system using data mining methods. *arXiv* **2013**, arXiv:1308.2831.
6. Cha, S.K.; Moraru, I.; Jang, J.; Truelove, J.; Brumley, D.; Andersen, D.G. SplitScreen: Enabling efficient, distributed malware detection. *J. Commun. Netw.* **2011**, *13*, 187–200. [CrossRef]
7. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Op-code sequences as representation of executables for data-mining-based unknown malware detection. *IET Inf. Sci.* **2013**, *231*, 64–82. [CrossRef]
8. Tabish, S.M.; Shafiq, M.Z.; Farooq, M. Malware detection using statistical analysis of byte-level file content. In Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, Paris, France, 28 June 2009; pp. 23–31.
9. Sharma, A.; Sahay, S.K.; Kumar, A. Improving the detection accuracy of unknown malware by partitioning the executables in groups. In *Advanced Computing and Communication Technologies, Proceedings of the 9th ICACCT, Panipat, India, 27–29 November 2015*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 421–431.
10. Ye, Y.; Chen, L.; Hou, S.; Hardy, W.; Li, X. DeepAM: A heterogeneous deep learning framework for intelligent malware detection. *Knowl. Inf. Syst.* **2018**, *54*, 265–285. [CrossRef]
11. Zhu, D.; Jin, H.; Yang, Y.; Wu, D.; Chen, W. DeepFlow: WEB-basedware detection by mining Android application for abnormal usage of sensitive data. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 438–443.
12. Huang, W.; Stokes, J.W. MtNet: A multi-task neural network for dynamic malware classification. In Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, 7–8 July 2016; pp. 399-418.
13. Sharma, S.; Rama Krishna, C.; Sahay, S.K. Detection of advanced malware by machine learning techniques. In *Soft Computing: Theories and Applications, Proceedings of the SoCTA2017*; Springer: Singapore, 2019; pp. 333–342.
14. Hussain, A.; Asif, M.; Ahmad, M.B.; Mahmood, T.; Raza, M.A. Malware detection using machine learning algorithms for windows platform. In Proceedings of the International Conference on Information Technology and Applications: ICITA 2021, Dubai, United Arab Emirates, 13–14 November 2021; pp. 619–632.
15. Kolter, J.Z.; Maloof, M.A. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.* **2006**, *7*, 2721–2744. [CrossRef]
16. Schultz, M.G.; Eskin, E.; Zadok, F.; Stolfo, S.J. Data mining methods for detection of new malicious executables. In Proceedings of the 2001 IEEE Symposium on Security and Privacy: SP 2001, Oakland, CA, USA, 14–16 May 2001; pp. 38–49.
17. Azeez, N.A.; Odufuwa, O.E.; Misra, S.; Oluranti, J.; Damaševičius, R. Windows PE malware detection using ensemble learning. *Informatics* **2021**, *8*, 10. [CrossRef]
18. Rathore, H.; Agarwal, S.; Sahay, S.K.; Sewak, M. Malware detection using machine learning and deep learning. In Proceedings of the International Conference on Big Data Analytics, Seattle, WA, USA, 21–22 November 2018; pp. 402–411.
19. Sewak, M.; Sahay, S.K.; Rathore, H. An investigation of a deep learning based malware detection system. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018; pp. 1–5.
20. Hardy, W.; Chen, L.; Hou, S.; Ye, Y.; Li, X. DL4MD: A deep learning framework for intelligent malware detection. In Proceedings of the International Conference on Data Science (ICDATA), Cochin, India, 23–25 August 2016; p. 61.
21. Lucas, K.; Sharif, M.; Bauer, L.; Reiter, M.K.; Shintre, S. Malware makeover: Breaking ml-based static analysis by modifying executable bytes. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, Hong Kong, China, 7–11 June 2021.
22. Coronel, F.; Barreno, N.; Muñoz, P.; Zabala-Blanco, D.; Onofa, N.; Flores-Calero, M. Web-based personal access control system using facial recognition with deep learning techniques. In Proceedings of the 2022 IEEE Colombian Conference on Communications and Computing (COLCOM), Cali, Colombia, 27–29 July 2022; pp. 1–6.
23. Singla, A.; Agrawal, R. DisDSS: A novel Web-based smart disaster management system for determining the nature of a social media message for decision-making using deep learning—Case study of COVID-19. *Glob. Knowl. Mem. Commun.* 2023, *ahead of print*. [CrossRef]
24. Ismail, N.A.; Chai, C.W.; Samma, H.; Salam, M.S.; Hasan, L.; Wahab, A.; Mohamed, F.; Leng, W.; Rohani, M.F. Web-based University Classroom Attendance System Based on Deep Learning Face Recognition. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 503–523.

25. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
26. Kumar, A. ClaMP (Classification of Malware with PE headers). Mendeley Data 2020, V1. Available online: https://doi.org/10.17632/xvyv59vwvz.1 (accessed on 25 July 2023).
27. Mauricio. Benign & Malicious PE Files. Kaggle Data 2018, V1. Available online: https://www.kaggle.com/datasets/amauricio/pe-files-malwares (accessed on 25 July 2023).
28. Yıldırım, E. MalwareDataSet. GitHub 2022, V1. Available online: https://github.com/emr4h/Malware-Detection-Using-Machine-Learning (accessed on 25 July 2023).
29. Dwyer, G.; Aggarwal, S.; Stouffer, J. *Flask: Building Python Web Services*; Packt Publishing: Birmingham, UK, 2017.