electronics

MDPI

Article

Ensemble Malware Classification System Using Deep Neural Networks

Barath Narayanan Narayanan [1,2,*] and Venkata Salini Priyamvada Davuluru [1,2]

[1] Department of Electrical and Computer Engineering, University of Dayton, Dayton, OH 45469, USA; davuluruv1@udayton.edu

[2] Sensors and Software Systems, University of Dayton Research Institute, Dayton, OH 45469, USA

* Correspondence: narayananb1@udayton.edu

Received: 1 April 2020; Accepted: 25 April 2020; Published: 27 April 2020

Abstract: With the advancement of technology, there is a growing need of classifying malware programs that could potentially harm any computer system and/or smaller devices. In this research, an ensemble classification system comprising convolutional and recurrent neural networks is proposed to distinguish malware programs. Microsoft's Malware Classification Challenge (BIG 2015) dataset with nine distinct classes is utilized for this study. This dataset contains an assembly file and a compiled file for each malware program. Compiled files are visualized as images and are classified using Convolutional Neural Networks (CNNs). Assembly files consist of machine language opcodes that are distinguished among classes using Long Short-Term Memory (LSTM) networks after converting them into sequences. In addition, features are extracted from these architectures (CNNs and LSTM) and are classified using a support vector machine or logistic regression. An accuracy of 97.2% is achieved using LSTM network for distinguishing assembly files, 99.4% using CNN architecture for classifying compiled files and an overall accuracy of 99.8% using the proposed ensemble approach thereby setting a new benchmark. An independent and automated classification system for assembly and/or compiled files provides the luxury to anti-malware industry experts to choose the type of system depending on their available computational resources.

Keywords: cyber security; malware classification; convolutional neural networks; recurrent neural networks; support vector machine; logistic regression

1. Introduction

Classifying malware programs into different categories based on their pattern has been a research area attracting great interest for several years [1]. Malware programs can either be present in the form of assembly files or binary files or even both in a computer or any other electronic device such as mobile phones and laptops. Anti-malware industries have remedial measures after associating a given malware program with a particular category. However, identifying a particular category of malware program can be difficult and extensive due to polymorphism and huge file size. Hackers introduce the concept of polymorphism to represent malware programs in different forms and sizes to make it difficult for the anti-malware industry to classify or identify such files. Computationally efficient Machine Learning (ML) methods to identify such patterns and malware programs are currently needed. Identifying features and providing insights into both assembly level and compiled files would be of valuable help for the anti-malware industry experts.

Several approaches have been studied in the literature to distinguish malware programs [2–22]. The most commonly used techniques include static [2–4], dynamic [5,6] and signature-based analysis. In addition, [7–10] presented various classification approaches for classifying malware programs after visualizing them as images using only compiled files. In [7], after engineering features using the

visualization technique, authors study the performance of traditional classification approaches using Support Vector Machine (SVM), k-Nearest Neighbors (kNN) and Artificial Neural Networks (ANNs). In [8], authors study the performance of autoencoders after visualizing malware programs as images. In [9], a multi-level architecture using both a traditional ML approach and an autoencoder-based approach is presented. A traditional ML approach is utilized to identify the minority class and an autoencoder is used to subcategorize all other classes. In [10], authors of this paper presented a Convolutional Neural Network (CNN) approach to classify malware patterns after visualizing them as images. In [10], the performance of a CNN is studied both as a feature extractor and a classification tool and the best performance is obtained by utilizing CNN as a feature extractor and SVM for classification. One of the striking advantages of visualizing a malware bytes file is that we could identify patterns with minimal memory consumption. In [12], computer viruses are visualized at an early stage utilizing Windows executable files with the help of self-organizing maps, without using virus specific signature information. In [13], dynamic analysis of program execution is presented using Ether hypervisor framework to monitor, process and then perform reverse engineering for compiled executables. In [14], results of various software analysis tools are brought together into a visual environment to support the triage and explore code vulnerabilities, thereby reducing the false positives in the voluminous data. In [15], quick analysis with visualization through treemaps and thread graphs is presented based on parameterized abstraction of detailed behavioral reports for detecting and classifying the maliciousness of software. In [16], an automated means to map the large binary objects is studied to classify regions using a multi-dimensional, information-theoretic approach. In [18], limitations of static and dynamic detection of malicious codes are studied, and a CNN-based detection model is employed. In [19], CNN is used for compiled files and Long-Short Term Memory (LSTM) is utilized for assembly files where malware files are classified using a stacking approach. Accuracies are obtained using the existing approaches and our approaches are compared and presented in the experimental results section. In [20], malware binaries are trained using deep learning models to classify each malware family. In [21], one-class classification is implemented that takes privileged information into account during the training phase to detect anomalies. In [22], authors present a gene sequence method of classification to distinguish the malware programs.

Even though so many algorithms have been proposed in the literature, very few research papers [19] have worked on combining both assembly level and compiled files. We extend our work from [10] for malware classification. In this paper, we present multiple approaches to classify malware programs. We present independent architectures for compiled and assembly files so that users can pick their choice based on the availability of files and computational resources. Another contribution of this paper is a novel approach of fusing both Natural Language Processing (NLP)-based approaches and image-based approaches into a single simple architecture. In this paper, we present a novel approach to utilizing CNNs and LSTMs as feature extractors instead of classification tools to combat class imbalance and limited training images present in the dataset. Representing such huge malware programs in a simple suite of features helps in the reduction of memory and computational time.

In this paper, three different approaches are presented and studied to classify malware programs based on different file formats: (a) CNN-based approach for classifying malware compiled files after visualizing them as images as implemented in [10], (b) Recurrent Neural Network (RNN)-based approach for classifying malware assembly files, (c) novel ensemble approach of combining the features extracted using (a) and (b) technique and later classifying them either using logistic regression or SVM. This type of approach helps in providing the end-user (anti-malware industry expert) in choosing the type of network of their choice. This type of approach also helps in choosing the type of network based on the files available as unique networks are presented for each domain. Some relevant information could be missing in either compiled or assembly level files for a given malware program, this could be addressed using our proposed approach. Our proposed approach helps in overcoming class imbalance issue present in the dataset without any data augmentation. An ensemble approach helps in taking advantage of both the domains and would help in providing valuable insights to the industry expert

on the correlation between the compiled and assembly level files. Not much research has been done on utilizing recurrent-based networks for assembly files which we believe, is an area of great interest. In addition, we compare the performance of these models with N-gram techniques using SVM for classification of assembly files. Furthermore, the ability to combine both sequential and visualization techniques using a logistic regression or SVM helps in achieving a reliable performance with minimal memory consumption.

In addition, the results are presented for publicly available BIG 2015 dataset [17] thereby serving as a benchmark for future research efforts. The remainder of this paper is organized as follows. Section 2 provides a brief description of the database that is employed for this research. Section 3 presents the training, testing and validation dataset distribution. Section 4 elucidates the CNN approach adopted for classification of compiled files along with the proposed RNN-based approach for classification of assembly files adopted in this paper. Section 4 also presents the proposed ensemble classification approach. Section 5 presents the experimental results obtained using the proposed methods. Finally, discussions are offered in Section 6.

2. Materials and Methods

As mentioned, a publicly available dataset provided in Kaggle by Microsoft for Malware Classification Challenge (BIG 2015) [17] is utilized for this research. This dataset is categorized into 9 different groups by anti-malware industry experts. In Kaggle, separate training and testing datasets are provided as a part of the competition. However, ground truth is not provided for the testing dataset and since the competition is closed, we cannot upload the results in order to estimate our performance. The performance of our technique is analyzed by performing hold-out validation solely using the training dataset provided by Kaggle. Training and testing distribution utilized for this research is provided in Section 3. Figure 1 shows the distribution of the training dataset of malware programs [10]. Table 1 displays their corresponding IDs and malware categories [10].

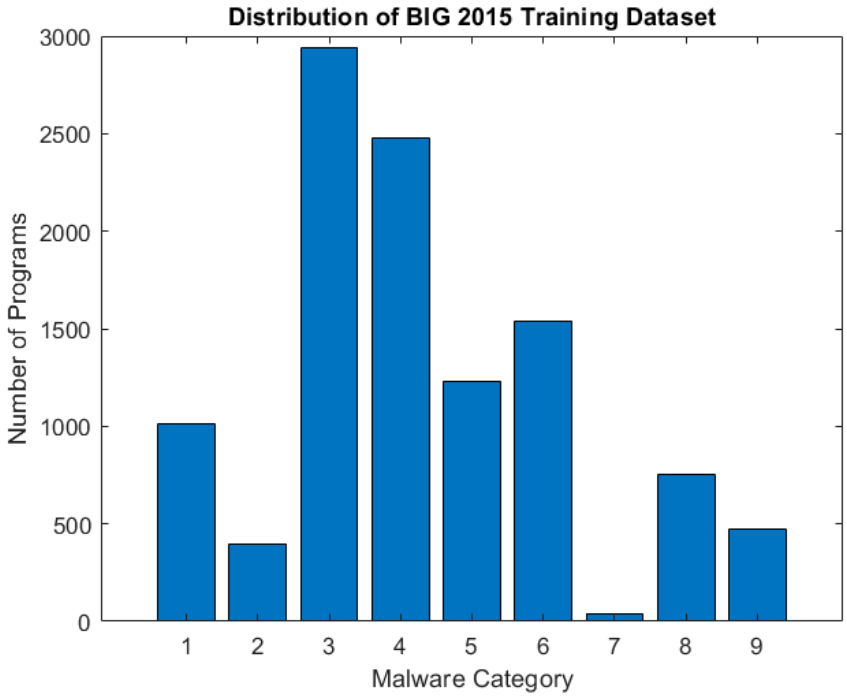

Figure 1. Malware Category Distribution in BIG 2015 Dataset [10].

Table 1. Malware categories and their class ID [10].

Class ID	Malware Category
1	Gatak
2	Kelihos_ver1
3	Kelihos_ver2
4	Lollipop
5	Obfuscator_ACY
6	Ramnit
7	Simda
8	Tracur
9	Vumdo

For each malware program, an assembly and a compiled file has been provided. Both set of files are preprocessed differently. At first, the compiled file is converted to images by converting every 2-digit hexadecimal code to its equivalent decimal number thereby achieving the range of grayscale image. These images are later resized as necessary for the CNN architectures implemented, as done in [10]. More details in regards to these preprocessing techniques are provided in [7,11]. Figure 2 presents visualization patterns obtained for each category [10].

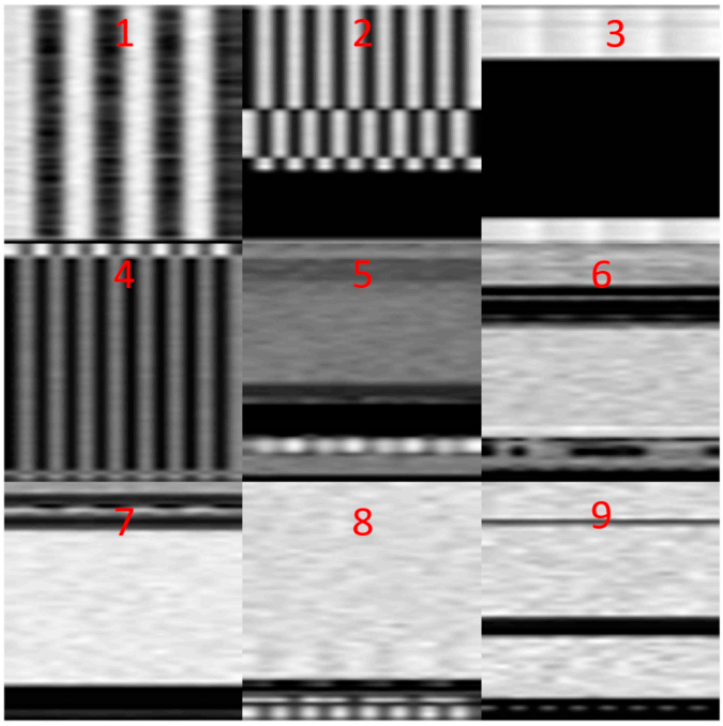

Figure 2. Visualization of Malware Programs [10].

The assembly files are preprocessed by extracting only the list of opcodes present in every file and ignoring other commands present in them. The order in which these opcodes are maintained are present in the file to preserve the sequential pattern present in these assembly files. There are about 483 unique opcodes present in all assembly files in the training dataset of Kaggle. Figure 3 presents the word cloud for each malware category solely based on assembly opcodes. Larger font indicates high frequency of the opcodes. These word clouds clearly indicate that different sets of opcodes are utilized in different frequencies for each malware category. This type of visualization would also assist the anti-malware experts to look for key opcodes for each malware category.

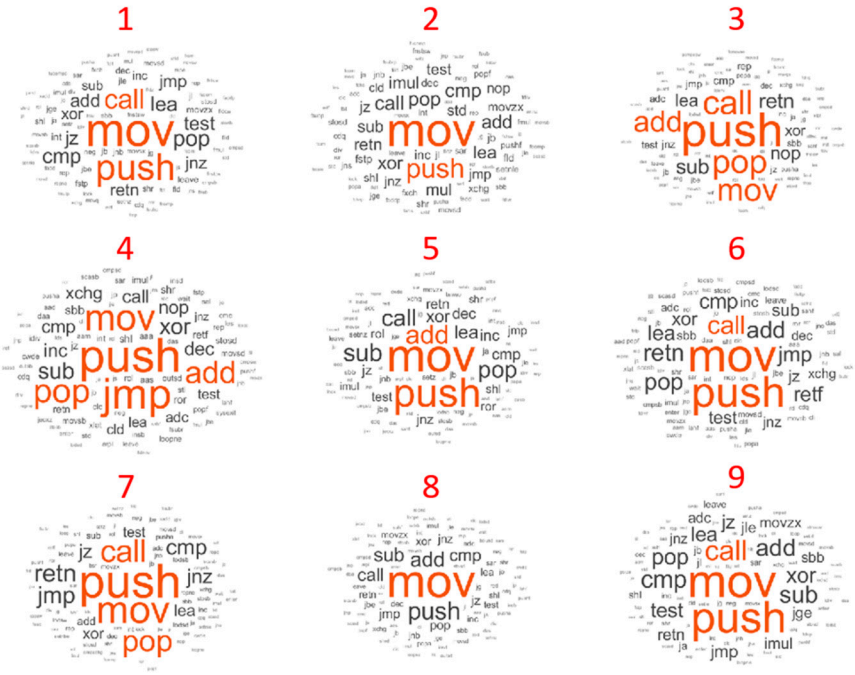

Figure 3. Word Cloud of Malware Programs.

Later, the opcodes are encoded in the entire training set to numeric indices. Figure 4 presents the lengths of documents present in the Kaggle training dataset after encoding. We study the performance of both the traditional N-gram approach with SVM and a custom RNN architecture.

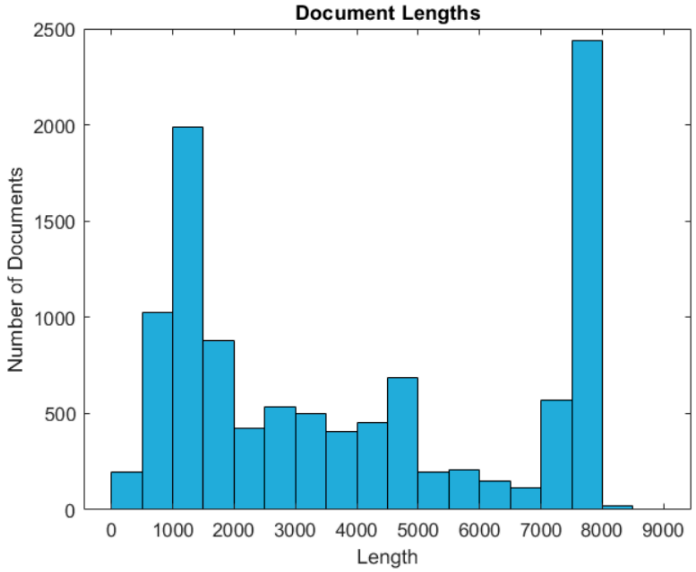

Figure 4. Length of assembly level files after opcode-based encoding.

All these documents are converted into sequences of 2000 words. Sequences with less than 2000 words are padded on the left with a value of 0 and the longer sequences are truncated. We believe that a sequential list of 2000 opcodes could be a sweet spot in recognizing the malware category with minimal memory consumption. The proposed algorithm achieves an accuracy of 97.2% for classification of assembly files, 99.4% for classification of binary files and the ensemble approach achieves an overall accuracy of 99.8%.

3. Dataset Distribution

As mentioned earlier, the training dataset provided in Kaggle as a part of this challenge is split into groups of 72%, 8% and 20% for training, validation and testing purposes, respectively, as done in [10]. The same set of cases are utilized throughout this research. Figure 5 and Table 2 shows the distribution of training, validation and testing datasets respectively [10].

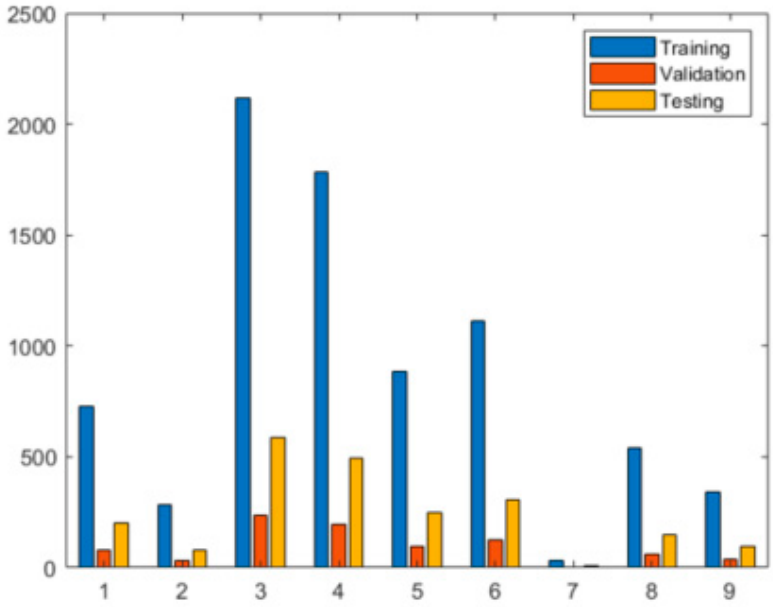

Figure 5. Distribution of training, validation and testing dataset [10].

Table 2. Training, validation and testing dataset composition.

Class ID	Number of Training Samples	Number of Validation Samples	Number of Testing Samples
1	729	81	203
2	286	32	80
3	2119	235	588
4	1784	198	496
5	884	98	246
6	1110	123	308
7	31	3	8
8	541	60	150
9	342	38	95

4. Classification Architecture

4.1. CNN Architecture

For classifying the malware compiled files after visualizing them as images, the architecture presented in [10] is adopted. In [10], several CNN architectures were presented and compared. CNN as feature extractor and SVM with linear kernel as the classifier achieved the highest accuracy, which is employed in this study. Features are extracted from the last fully-connected layer of several well-established CNN-based architectures which includes AlexNet [23], ResNet [24] and VGG-16 [25] after training using our dataset. In addition, a set of features are extracted from a simple CNN architecture. In total, 36 features are extracted (9 from each architecture) and are later classified using SVM with linear kernel. This type of architecture helps in addressing class imbalance problem associated with the BIG 2015 dataset. CNN weights are already learned, there is no need to retrain the dataset, thereby making it easier for its deployment. We adopt the same architecture for classifying compiled files in this research.

4.2. RNN Architecture

To classify assembly files, LSTM network, a type of RNN is proposed. After preprocessing the assembly files as mentioned in Section 2, preprocessed files serve as the input to the LSTM architecture [26]. LSTM neural networks have been highly effective for natural language processing classification problems and help in identifying patterns in sequences, hence the same is adopted in this research. Figure 6 presents the architecture adopted in this paper for classification of assembly level files.

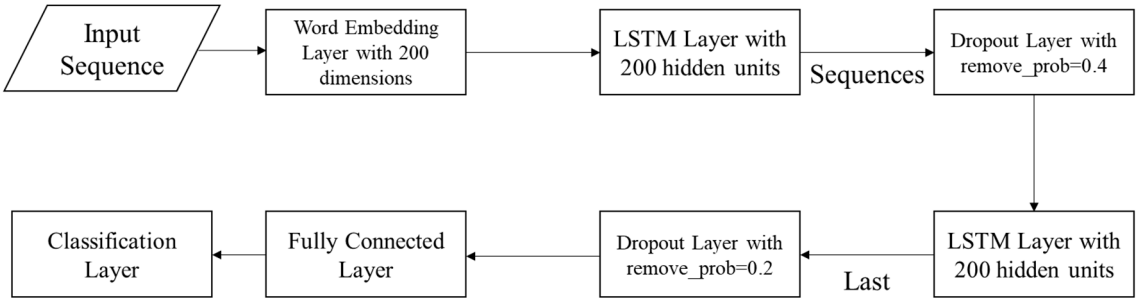

Figure 6. Proposed recurrent neural network (RNN) architecture.

4.3. Ensemble Architecture

In this subsection, we present our novel approach of fusing both RNN- and CNN-based architectures for the classification of malware programs. This type of architecture is implemented to have a single representation for a given malware program. In addition, some malware programs do not contain sufficient information/data in either assembly or compiled files; this could be tackled using our approach. For the ensemble approach, we extract features from the last fully connected layer of our trained models for CNN and RNN, respectively. We extract 9 features from each of those architectures compiling a suite of 45 in total. The extraction of features is adopted as it has been proven to be an effective overcome limited training data and class imbalance issue for malware classification [10]. These features represent the malware programs both in terms of assembly and compiled files. These extracted features are classified using logistic regression and SVM for our study. Figure 7 presents the block diagram of this proposed approach. This type of architecture not only assists in representing malware program containing assembly level and compiled files, it also assists in overcoming the class imbalance issue present in the BIG 2015 dataset. This type of architecture can be adapted to various applications containing different file types to represent a training sample.

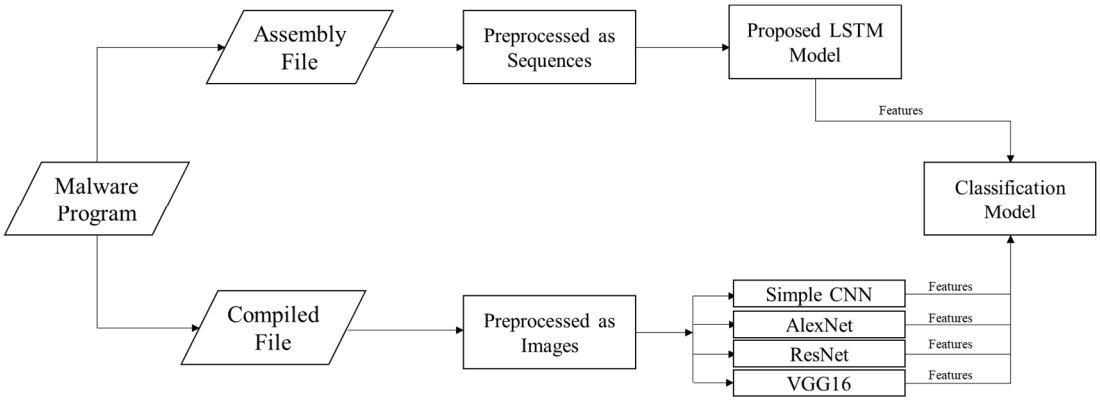

Figure 7. Proposed Ensemble Architecture.

5. Results

In this section, results obtained for assembly files, compiled files and their combination using our proposed approaches are presented. At first, results obtained after visualizing our compiled files as images, extracting features using CNN and later classifying them using SVM [10], are presented. Figure 8 presents the confusion matrix obtained using this approach. An overall accuracy of 99.4% is achieved for classifying malware programs solely based on compiled files.

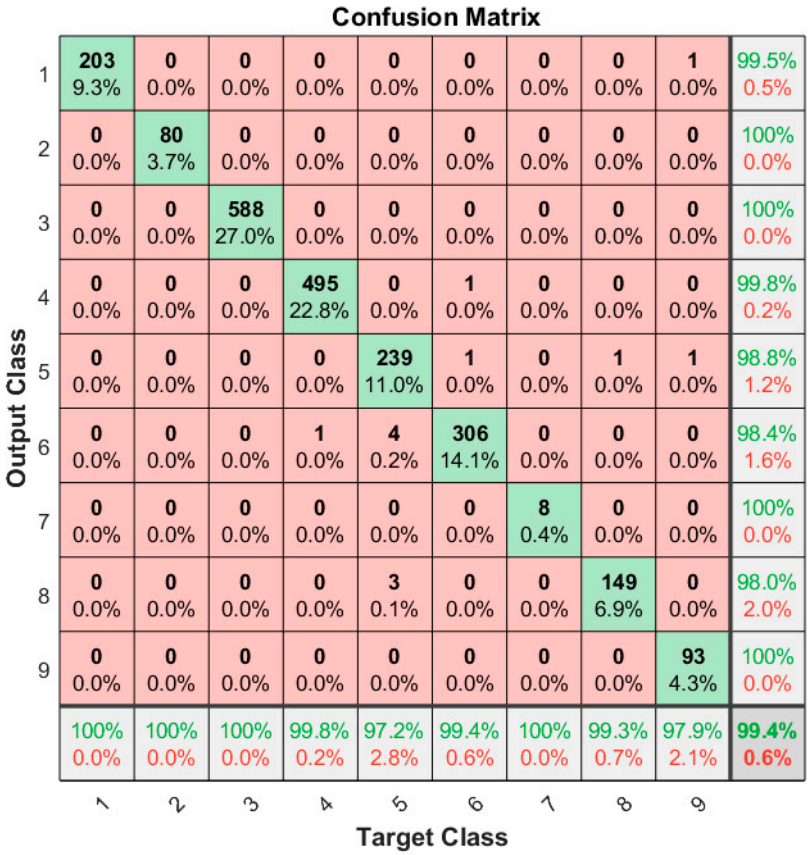

Figure 8. Confusion matrix obtained using a convolutional neural network (CNN) for classification of compiled files [10].

Figure 9 presents the confusion matrix obtained using N-gram technique (N = 4) using SVM with linear kernel for classification. Table 3 presents the results obtained using different values of "N" with this approach. Best results are obtained with 4-g technique for classification of assembly files.

Figure 9. Confusion matrix obtained using CNN for classification of compiled files [10].

Table 3. Performance comparison of N-gram approach using a support vector machine (SVM) with linear kernel.

N	Accuracy (%)
2	91.2
3	95.6
4	96.2
5	95.7

Figure 10 presents the confusion matrix obtained using the proposed LSTM architecture for classifying malware programs solely based on assembly files. An overall accuracy of 97.2% is achieved using the proposed approach. Figures 9 and 10 clearly indicate that proposed LSTM-based architecture outperforms the N-gram approach.

Results obtained using our ensemble approach are presented for both logistic regression and SVM. A combination of 45 features is obtained (as mentioned in Figure 6) and the performance of logistic regression and SVM for classification is studied. Our proposed ensemble approach significantly outperforms the independent architectures for assembly level and compiled files, respectively. Representing such huge malware programs in terms of 45 features significantly reduces the computational time and complexity. In addition, representing both assembly level and compiled files in terms of 45 features helps in overcoming missing information present in either of the file types. Figures 11 and 12 present the results obtained using logistic regression and SVM respectively. Table 4 presents the performance of various classification approaches presented for the BIG 2015 dataset [22]. This performance includes principal component analysis (PCA) using traditional classification approaches [7], autoencoders [8], CNN for classifying compiled files [10], the random forest approach [20], one-class SVM approach, strand gene sequence [21] and our proposed approaches.

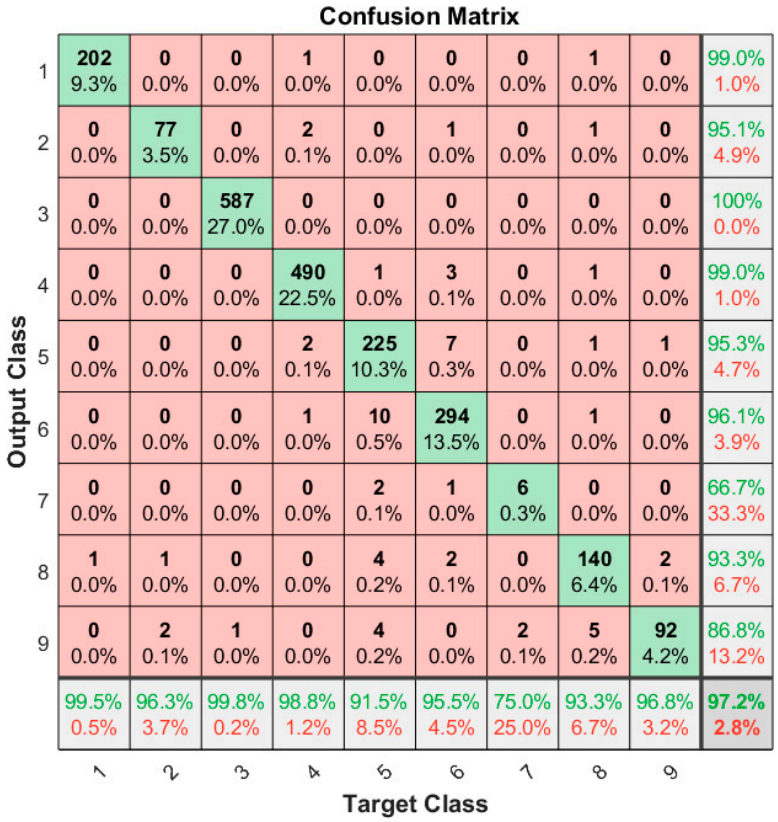

Figure 10. Confusion matrix obtained using proposed RNN for classification of assembly files.

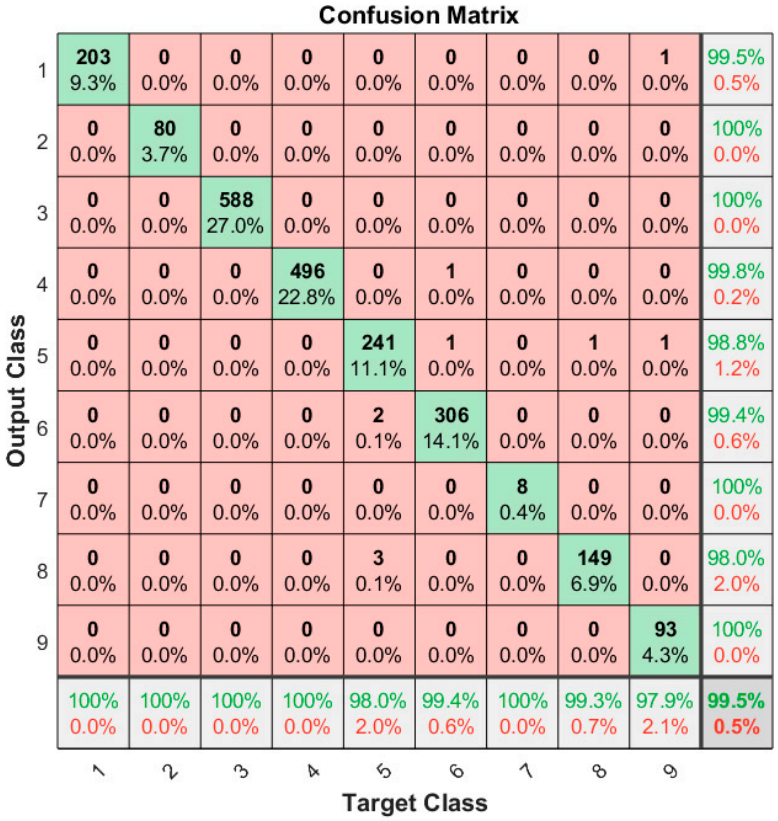

Figure 11. Confusion matrix obtained using ensemble approach with logistic regression.

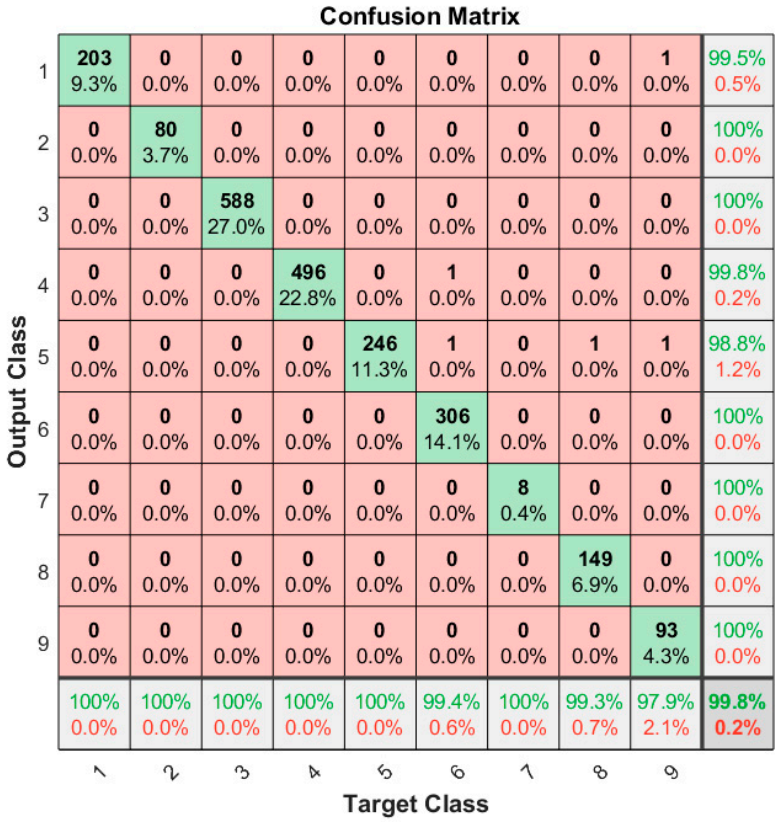

Figure 12. Confusion matrix obtained using ensemble approach with SVM.

Table 4. Performance comparison of various malware classification algorithms for BIG 2015 dataset.

Algorithm	Accuracy (%)
One-Class SVM [21]	92
Random Forest [20]	95.6
N-gram with SVM with only Assembly Files	**96.2**
PCA and kNN [7]	96.6
Proposed LSTM with only Assembly Files	**97.2**
Strand Gene Sequence [21]	98.59
Autoencoders [8] [1]	99.1
MalNet [19]	99.3
Our CNN Feature Extraction and SVM [10]	**99.4**
Proposed Ensemble Approach using Logistic Regression	**99.5**
Proposed Ensemble Approach using SVM	**99.8**

[1] Note: [8] removed Simda malware programs from analysis due to its limited availability. Results obtained using our proposed approaches are presented in bold.

6. Discussion

In this paper, multiple classification approaches are presented for distinguishing malware programs depending on the type of data available. Results clearly indicate that extracting features using CNN and classification using SVM provides the best performance for the classification of compiled files. An accuracy of 99.4% is achieved for classifying malware programs solely utilizing compiled files. In this research, a novel approach for classifying assembly files using a simple LSTM network is presented. An accuracy of 97.2% is achieved for the classification of malware programs solely based on assembly files, thereby setting a new benchmark. A simple LSTM approach outperforms the N-gram approach with SVM.

Our proposed ensemble approach provided a further boost in performance. Extracting features from our proposed CNN and RNN and later classifying them using logistic regression and SVM provided a further improvement in performance. Extracting a total of 45 features from all these networks helped in distinguishing the malware programs effectively. An overall accuracy of 99.5% and 99.8% is achieved using logistic regression and SVM, respectively. Table 4 clearly indicates that our proposed approach outperforms several other existing approaches. SVM coupled with 45 features extracted using CNN and RNN provided the best performance. Representing malware programs both in terms of compiled and assembly level files helped in overcoming a lack of information present in either of those file types. Representing such huge malware programs in terms of a simple suite of 45 features helps in reducing data complexity and computational resources. Extracting features from each of these architectures takes about 50 milliseconds for each malware program after training. This computational speed test was conducted on NVIDIA GeForce GTX-1070. The study of algorithms in terms of performance and type of files helps the anti-malware industry experts to choose the algorithm based on their needs. This type of independent architecture for each file type also helps in retraining any particular architecture depending on the new set of data collected. Any network can be easily modified without affecting other architectures utilized for feature extraction. This type of automated detection of malware programs would be valuable for the anti-malware industry.

Author Contributions: Conceptualization, V.S.P.D. and B.N.N.; methodology, V.S.P.D. and B.N.N.; software, V.S.P.D.; validation, V.S.P.D., B.N.N.; formal analysis, V.S.P.D.; resources, V.S.P.D. and B.N.N.; writing—original draft preparation, V.S.P.D. and B.N.N.; writing—review and editing, V.S.P.D., B.N.N.; visualization, V.S.P.D.; supervision, B.N.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We render our sincere thanks to Microsoft Corp. for making the Malware Classification Challenge (BIG 2015) dataset publicly available.

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

References

1. Symantec Internet Security Threat Report. 2019. Available online: https://www-west.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf (accessed on 25 April 2020).
2. Tian, R.; Batten, L.M.; Versteeg, S.C. Function length as a tool for malware classification. In Proceedings of the 2008 3rd International Conference on Malicious and Unwanted Software (MALWARE), Fairfax, VI, USA, 7–8 October 2008; pp. 69–76.
3. Karim, M.E.; Walenstein, A.; Lakhotia, A.; Parida, L. Malware phylogeny generation using permutations of code. *J. Comput. Virol.* **2005**, *1*, 13–23. [CrossRef]
4. Kolter, J.Z.; Maloof, M.A. Learning to detect malicious executables in the wild. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM), Seattle, WA, USA, 22–25 August 2004; pp. 470–478.
5. Park, Y.; Reeves, D.; Mulukutla, V.; Sundaravel, B. Fast malware classification by automated behavioral graph matching. In Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research (ACM), Oak Ridge, TN, USA, 21–23 April 2010; p. 45.
6. Cesare, S.; Yang, X. Classification of malware using structured control flow. In Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing, Darlinghurst, Australia, 18 January 2010; Australian Computer Society, Inc.: Darlinghurst, Australia, 2010; Volume 107, pp. 61–70.
7. Narayanan, B.N.; Djaneye-Boundjou, O.; Kebede, T.M. Performance analysis of machine learning and pattern recognition algorithms for malware classification. In Proceedings of the 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), Dayton, OH, USA, 25–29 July 2016; pp. 338–342.
8. Kebede, T.M.; Djaneye-Boundjou, O.; Narayanan, B.N.; Ralescu, A.; Kapp, D. Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (big 2015) dataset. In Proceedings of the 2017 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 27–30 June 2017; pp. 70–75.

9. Messay-Kebede, T.; Narayanan, B.N.; Djaneye-Boundjou, O. Combination of Traditional and Deep Learning based Architectures to Overcome Class Imbalance and its Application to Malware Classification. In Proceedings of the NAECON 2018-IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, 23–26 July 2018; pp. 73–77.
10. Davuluru, V.S.P.; Narayanan, B.N.; Balster, E.J. Convolutional Neural Networks as Classification Tools and Feature Extractors for Distinguishing Malware Programs. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019; pp. 273–278.
11. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security (ACM), Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.
12. Yoo, I. Visualizing windows executable viruses using self-organizing maps. In Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, Washington, DC, USA, 29 October 2004; pp. 82–89.
13. Quist, D.A.; Liebrock, L.M. Visualizing compiled executables for malware analysis. In Proceedings of the 2009 6th International Workshop on Visualization for Cyber Security (IEEE), Atlantic City, NJ, USA, 11 October 2009; pp. 27–32.
14. Goodall, J.R.; Radwan, H.; Halseth, L. Visual analysis of code security. In Proceedings of the Seventh International Symposium on Visualization for Cyber Security (ACM), Ottawa, ON, Canada, 14 September 2010; pp. 46–51.
15. Trinius, P.; Holz, T.; Göbel, J.; Freiling, F.C. Visual analysis of malware behavior using treemaps and thread graphs. In Proceedings of the 2009 6th International Workshop on Visualization for Cyber Security, Atlantic City, NJ, USA, 11 October 2009; pp. 33–38.
16. Conti, G.; Bratus, S.; Shubina, A.; Sangster, B.; Ragsdale, R.; Supan, M.; Lichtenberg, A.; Perez-Alemany, R. Automated mapping of large binary objects using primitive fragment type classification. *Digit. Investig.* **2010**, *7*, S3–S12. [CrossRef]
17. Kaggle BIG 2015 Dataset. Available online: https://www.kaggle.com/c/malware-classification (accessed on 11 November 2019).
18. Ronen, R.; Radu, M.; Feuerstein, C.; Yom-Tov, E.; Ahmadi, M. Microsoft malware classification challenge. *arXiv* **2018**, arXiv:1802.10135. Available online: https://arxiv.org/abs/1802.10135 (accessed on 25 April 2020).
19. Yan, J.; Qi, Y.; Rao, Q. Detecting malware with an ensemble method based on deep neural network. *Secur. Commun. Netw.* **2018**. [CrossRef]
20. Garcia, F.C.C.; Muga, I.I.; Felix, P. Random forest for malware classification. *arXiv* **2016**, arXiv:1609.07770. Available online: https://arxiv.org/abs/1609.07770 (accessed on 25 April 2020).
21. Burnaev, E.; Smolyakov, D. One-class SVM with privileged information and its application to malware detection. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 273–280.
22. Drew, J.; Moore, T.; Hahsler, M. Polymorphic malware detection using sequence classification methods. In Proceedings of the 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 22–26 May 2016; pp. 81–87.
23. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 3–8 December 2012; pp. 1097–1105.
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
25. Simonyan, K.; Andrew, Z. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556. Available online: https://arxiv.org/abs/1409.1556 (accessed on 25 April 2020).
26. Sundermeyer, M.; Ralf, S.; Hermann, N. LSTM neural networks for language modeling. In Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association, Portland, OR, USA, 9–13 September 2012.