

Comparison of Deep Learning and the Classical Machine Learning Algorithm for the Malware Detection

Mohit Sewak
BITS, Pilani,
Department of CS & IS,
Goa Campus, India
p20150023@goa.bits-pilani.ac.in

Sanjay K. Sahay
BITS, Pilani,
Department of CS & IS,
Goa Campus, India
ssahay@goa.bits-pilani.ac.in

Hemant Rathore
BITS, Pilani,
Department of CS & IS,
Goa Campus, India
hemantr@goa.bits-pilani.ac.in

Abstract—Recently, Deep Learning has been showing promising results in various Artificial Intelligence applications like image recognition, natural language processing, language modeling, neural machine translation, etc. Although, in general, it is computationally more expensive as compared to classical machine learning techniques, their results are found to be more effective in some cases. Therefore, in this paper, we investigated and compared one of the Deep Learning Architecture called Deep Neural Network (DNN) with the classical Random Forest (RF) machine learning algorithm for the malware classification. We studied the performance of the classical RF and DNN with 2, 4 & 7 layers architectures with the four different feature sets, and found that irrespective of the features inputs, the classical RF accuracy outperforms the DNN.

Keywords—Malware, Deep Learning, Machine Learning, Auto Encoders, Deep Neural Networks.

I. INTRODUCTION

In today's world, information is one of the most valuable assets, but there is a major threat to it by the evolving sophisticated malware (**malicious software**). The first malware was created for fun, but now its a profit-driven industry [1] and in the last couple of years advanced malware uses complex obfuscation methods viz. control/data flow permutation, compression, heap spray, etc. [2] to evade the detection techniques. Also, there is an exponential increase in the number of malware released every year. To detect these malware, various techniques are proposed in the past. These methods range from the early day signature-based detection to Machine Learning techniques [3]. However, it appears that proposed approaches are not sufficient to detect the unknown sophisticated malware. Thus researchers are continuously searching for better techniques to detect the advanced malware and have started applying machine learning techniques for malware detection. These algorithms can detect obfuscated malware and have potential to handle both complexity and scale problem like data pre-processing, feature extraction/selection, model creation and testing. In this Santos et. al claims to obtain an accuracy of $\sim 96\%$ using opcode frequency as features [4]. Later on, researchers also applied file-size based segmentation on opcode frequency and improved the average accuracy using different Machine Learning models and claimed that Random Forest can provide the accuracy up to $\sim 98\%$ with a False

Positive Rate (FPR) of $\sim 1.07\%$ [5]. For the analysis they used the benchmark malware samples from the Malicia project [6].

Recently, Deep Learning has been showing promising results in various Artificial Intelligence applications like image recognition, natural language processing, language modeling, neural machine translation, etc. [7]. Therefore, in this paper, we compared the effectiveness of a Deep Learning Architecture called Deep Neural Network (DNN) with the classical Random Forest (RF) machine learning algorithm for the malware classification. We did not use the segmentation method as used in the above work. Instead, we used different feature selection/extraction techniques and achieved a much better accuracy ($\sim 99\%$) using Random Forest with an even better FPR ($\sim 0.24\%$). Our analysis indicates that Deep Learning based architectures such as Auto-Encoders for feature extraction combining with Deep Neural Networks for the classification does not give statistically better accuracy as compared to state-of-art machine learning technique combinations like Variance Threshold (for feature selection) and Random Forest (for classification) on Malicia data set. Rest of the article is organized as follows. Section 2, in brief, discusses the related work done in the malware classification. Section 3 explains the data preprocessing, feature extraction and selection approach for comparing the model. Section 4 compares the results of DNN (2, 4 and 7 layers) with classical RF classifier and Section 5 finally contains the conclusion of the article.

II. RELATED WORK

There are two basic approaches to identify malware from the benign programs, namely static analysis, and dynamic analysis. In the static approach without executing the program features are extracted from the file or its associated metadata, and then using the selected/extracted features (e.g., opcode frequency [5], strings & byte sequence [8], function length [9], API Calls [10], etc.) the detection algorithm is applied for the classification. Whereas in the dynamic analysis, the program is executed, and then important behavioral aspects and other parameter are captured as features on which the detection algorithm is applied. In addition, there also exists approaches where different features from both the static and dynamic analysis are combined to form a Hybrid approach [11]. Bilar

[12] directly used op-code frequency distribution for malware detection whereas Karim et al. [13] used op-code sequences and permutations for the same. Ashu et al. applied file-size based grouping and opcode frequency as features to improved the average accuracy with different Machine Learning models with Malicia data set [6], and obtained accuracy up to 98%.

For the classification of malware by the supervised machine learning, typically a binary model is developed to differentiate malware and benign programs, and sometimes multinomial models are also constructed for distinguishing samples across the different malware families. In this some popular supervised machine learning algorithm used for malware classification are Naive Bayes [14], Support Vector Machine [4], [14], Decision tree [14], Iterative Dichotomiser 3, J48 [5], Hidden Markov Models, Random Forest [5], AdaBoost, Instance-Based Learning Algorithms [14] and TF-IDF [4], [14]. Also, for feature extraction/selection/dimension reduction algorithms can be also used with supervised learning to boost its performance. The popular way of feature extraction is to use an unsupervised learning algorithm for making better representative features at the supervised learning stage, e.g., feature selection methods such as hierarchical, unary variable removal, Goodness evaluator, and Weighted Term Frequency [4]. However, if feature extraction is not done properly, then it may result in lesser accuracy, higher false positive rates, etc. Due to these shortcomings, Deep Learning based techniques are gaining a lot of traction for the feature extraction and otherwise also. Recently a lot of efforts has been focused and achieved success with different Deep Learning techniques like Deep Belief Networks [15], Deep Neural Networks [16], Recurrent Neural Networks (RNN) (its variants like Long Short-Term Memory (LSTM) & Gated Recurrent Unit (GRU)) [17] and combination of Recurrent and Convolutional Neural Networks (CNN) [18] for supervised learning and classification. Similarly, for optimal feature extraction Deep Belief Networks (DBN) and (Stacked) Auto-Encoders (SAE/ AE), and RNN based Auto-Encoders (RNN-AE) [10] have been used.

III. DATA PRE-PROCESSING AND FEATURE SELECTION/ EXTRACTION

For the investigation of the popular Random Forest and Deep Neural Networks with 2, 4, 7 layers, we use the 11,308 malicious files downloaded from benchmark Malicia project (one of author posses the dataset [5]) with different sets of features for the classification of malware. For the analysis, we collected 2,819 benign files (cross-verified from the 'virustotal.com' [19] file scanning service) from different Windows systems. As the number of benign and malicious files are not comparable, we used Adaptive Synthetic (ADASYN) [20] technique to alleviate the class imbalance problem. Now using the Linux *objdump* utility, we extracted the opcodes of all the executables and made a cumulative list of all the unique opcodes (*master opcode list*). The extracted unique opcodes are mapped with an integer for further analysis. Then we computed the frequency of each opcode of every executable, and ordered them as per the master opcode list for the comparison of the classifiers under investigation.

The flowchart 1 shows our approach to classify the malware using Random Forest, 2, 4, and 7 layers Deep Neural Network with the different sets of feature viz. by

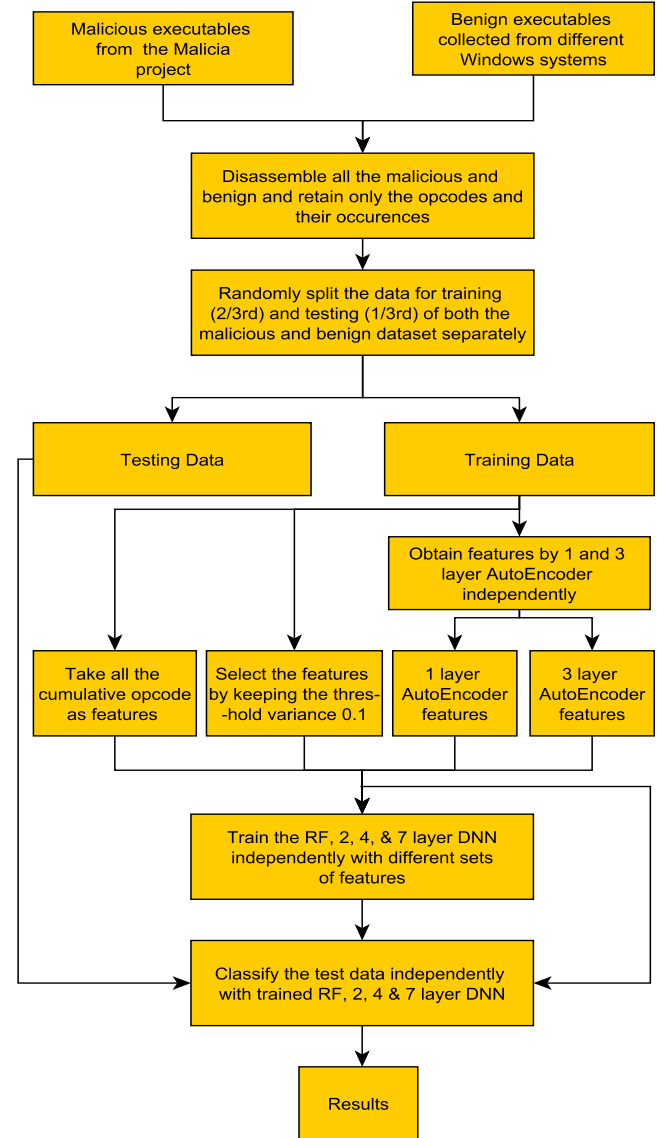


Fig. 1. Flowchart for the classification of malware by RF, 2, 4, 7 layers DNN with different sets of features.

- 1) taking all the cumulative opcodes, i.e., all the 1613 opcodes occurrence has been used for the classification.
- 2) selecting the features by keeping the variance threshold (VT) to 0.1, in this case, 616 opcodes occurrence has been used for the classification.
- 3) extracting the features by AutoEncoder with one layer (AE 1L) and three layers (AE 3L). In this input layer takes all the 1613 opcodes occurrence, and output layer has been set to 32 for both AE 1L and AE 3L.

Now we randomly split the dataset (containing only opcode occurrence of every executable) for the training and testing of the malicious and benign dataset separately (a conservative side as per the suggested norms to ensure optimal performance [21]) in the ratio of 2:1.

IV. COMPARISON OF DEEP LEARNING AND AN EFFECTIVE MACHINE LEARNING RANDOM FOREST ALGORITHM FOR THE MALWARE DETECTION

To compare the most popular and effective Machine Learning Random Forest classifier with Deep Learning approaches, we had chosen Deep Neural Network with 2, 4, and 7 layers. For the Random Forest, we use the Python library Sci-Kit Learn [22], for DNN and AE we use the Python library Keras [23] with TensorFlow [24] back-end. In DNN, three different layers (2, 4 and 7) architectures have been chosen to understand the performance of the DNN with the increase in the number of hidden layers. Here all the DNN layers use Exponential Linear Units (ELU) [25] as activation function, except for the last layer, i.e., prediction layer uses Sigmoid activation function, and the model are trained by setting the dropout rate to 0.1 with a batch size of 64. Also, DNN uses Adam [26] optimizer and binary cross-entropy ($H(p, q)$) [27] loss function for the distribution p (actual class) and q (probability of the predicted class) over a set of events (x) given as:

$$H(p, q) = - \sum_x p(x) \log q(x)$$

To compare the performance of the Random Forest with 100 trees in the forest, 2, 4, 7 layers DNN, we computed the True Positive Rate (TPR) / Sensitivity / Recall / Hit Rate, True Negative Rate (TNR) / Specificity, Positive Predictive Value (PPV) / Precision and Accuracy (Acc.) defined as [28]:

$$\text{TPR} = \frac{TP}{TM}; \quad \text{TNR} = \frac{TN}{TB}; \quad \text{PPV} = \frac{TP}{TP + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{TM + TB}$$

where,

- TP \longrightarrow True positives; the number of malware instances correctly classified.
- TN \longrightarrow True negatives; the number of benign instances correctly classified.
- FP \longrightarrow False positives; the number of benign instances wrongly detected as malware.
- TM \longrightarrow Total number of malware instances.
- TB \longrightarrow True number of benign instances.

From the investigation (Table I), we found that overall the Random Forest's results are better than the 2, 4, 7 layers DNN for all the four different sets of features viz. None (i.e., all the cumulative opcodes taken as the feature), VT, AE-1L, and AE-3L. However, if compared between all the DNN, seven layers DNN performance is the best (99.21% accuracy).

In the classification, features inputs to the classifier play a vital role. Therefore in terms of input features the analysis shows that classical RF performance is more or less same with None or VT features. DNN-2L gives the best accuracy with the features obtained by VT, whereas DNN with 4 layers gives the highest accuracy with AE-1L. However, among the DNN's (with different features inputs) AE-1L provides the maximum

Classifiers	Features	Acc.	TPR	TNR	PPV
RF	None	0.9974	0.9948	1.0000	1.0000
DNN 2L	None	0.9779	0.9633	0.9926	0.9924
DNN 4L	None	0.9742	0.9538	0.9948	0.9946
DNN 7L	None	0.9615	0.9905	0.932	0.9366
RF	VT	0.9978	0.9959	0.9997	0.9997
DNN 2L	VT	0.9884	0.9832	0.9937	0.9937
DNN 4L	VT	0.9869	0.9796	0.9942	0.9942
DNN 7L	VT	0.9620	0.9889	0.9348	0.9389
RF	AE 1L	0.9941	0.9886	0.9997	0.9997
DNN 2L	AE 1L	0.9695	0.9457	0.9937	0.9934
DNN 4L	AE 1L	0.9899	0.9829	0.9970	0.9970
DNN 7L	AE 1L	0.9921	0.9861	0.9981	0.9981
RF	AE 3L	0.9936	0.9872	1.0000	1.0000
DNN 2L	AE 3L	0.9625	0.9375	0.9879	0.9874
DNN 4L	AE 3L	0.9716	0.9861	0.9568	0.9585
DNN 7L	AE 3L	0.9360	0.8797	0.9931	0.9923

TABLE I. RESULTS WITH MULTIPLE FEATURE METHODS AND CLASSIFICATION MODELS

accuracy (99.21%) but if the feature input is from AE-3L, then there is ~6% decrease in the classification accuracy.

V. CONCLUSION

In this paper, we compare the performance of the classical machine learning Random Forest techniques and Deep Neural Networks (with multiple architectures) for the classification of malware with different sets of features. We found that overall classical ML algorithm Random Forest outperform the different layers architecture of DNN. The best accuracy obtained by RF and DNN are 99.78% and 99.21% with feature set obtained by VT and AE-1L respectively. This indicates that Deep Learning based architectures such as Auto-Encoders (used for feature extraction) and when combined with Deep Neural Networks (for the classification) may be an overkill the Malicia Dataset because its obfuscation technique may not be too complex to predict the malware using opcode frequency as a feature. Therefore it does not provide statistically better accuracy as compared to classical machine learning RF. However, further investigation is required to understand the effectiveness of Deep Learning techniques like RNN, LSTM, ESN, etc. coupled with more advanced data processing and feature extraction methods, and the work in this direction is in progress.

REFERENCES

- [1] Security software - statistics & facts. <https://www.statista.com/topics/2208/security-software>, 2016, Date last accessed 21-March-2018.
- [2] Ilsun You and Kangbin Yim. Malware obfuscation techniques: A brief survey. In *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, pages 297–300, November 2010.
- [3] Ashu Sharma and S. K. Sahay. Evolution and detection of polymorphic and metamorphic malwares: A survey. *International Journal of Computer Applications*, 90(2):7–11, March 2014.
- [4] Igor Santos, Felix Brezo, Xabier Ugarte-Pedrero, and Pablo G. Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, 231:64 – 82, 2013. Data Mining for Information Security.
- [5] Ashu Sharma and Sanjay Kumar Sahay. An effective approach for classification of advanced malware with high accuracy. *International Journal of Security and Its Applications*, 10(4), 2016.
- [6] Malicia project. <http://malicia-project.com>, 2012, Date last accessed 15-July-2014.
- [7] Mohit Sewak, Md. Rezaul Karim, and Pradeep Pujari. *Practical Convolutional Neural Network Models*. Packt Publishing Ltd., 2018.

- [8] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, pages 38–49, 2001.
- [9] R. Tian, L. M. Batten, and S. C. Versteeg. Function length as a tool for malware classification. In *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, pages 69–76, October 2008.
- [10] Xin Wang and Siu-Ming Yiu. A multi-task learning model for malware classification with useful file access pattern from api call sequence. *CoRR*, abs/1610.05945, 2016.
- [11] Mansour Ahmadi, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, and Giorgio Giacinto. Novel feature extraction, selection and fusion for effective malware family classification. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY '16*, pages 183–194, New York, NY, USA, 2016. ACM.
- [12] Daniel Bilal. Opcodes as predictor for malware. *Int. J. Electron. Secur. Digit. Forensic*, 1(2):156–168, January 2007.
- [13] Md. Enamul Karim, Andrew Walenstein, Arun Lakhotia, and Laxmi Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1:13–23, 2005.
- [14] Jeremy Z. Kolter and Marcus A. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pages 470–478, New York, NY, USA, 2004. ACM.
- [15] O. E. David and N. S. Netanyahu. Deepsign: Deep learning for automatic malware signature generation and classification. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2015.
- [16] J. Saxe and K. Berlin. Deep neural network based malware detection using two dimensional binary program features. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 11–20, October 2015.
- [17] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1916–1920, April 2015.
- [18] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. Deep learning for classification of malware system call sequences. In Byeong Ho Kang and Quan Bai, editors, *AI 2016: Advances in Artificial Intelligence*, pages 137–149, Cham, 2016. Springer International Publishing.
- [19] Virustotal. <http://virustotal.com>, 2015, Date last accessed 25-July-2014.
- [20] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, June 2008.
- [21] Isabelle Guyon. A scaling law for the validation-set training-set size ratio. In *AT & T Bell Laboratories*, 1997.
- [22] Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.
- [23] François Chollet et al. Keras. <https://keras.io>, 2015, Date last accessed 23-March-2017.
- [24] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Date last accessed 27-March-2017. Software available from tensorflow.org.
- [25] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.
- [26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [27] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, February 2005.
- [28] D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.