Big Data: Deep Learning for detecting Malware

Emmanuel Masabo
Makerere University
Kampala, Uganda
masabem@gmail.com

Kyanda Swaib Kaawaase
Makerere University
Kampala, Uganda
kswaibk@cis.mak.ac.ug

Julianne Sansa-Otim
Makerere University
Kampala, Uganda
sansa@cit.ac.ug

ABSTRACT

Malicious software, commonly known as malware are constantly getting smarter with the capabilities of undergoing self-modifications. They are produced in big numbers and widely deployed very fast through the Internet-capable devices. This is therefore a big data problem and remains challenging in the research community. Existing detection methods should be enhanced in order to effectively deal with today's malware. In this paper, we propose a novel real-time monitoring, analysis and detection approach that is achieved by applying big data analytics and machine learning in the development of a general detection model. The learnings achieved through big data render machine learning more efficient. Using the deep learning approach, we designed and developed a scalable detection model that brings improvement to the existing solutions. Our experiments achieved an accuracy of 97% and ROC of 0.99.

CCS CONCEPTS

• **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; *Malware mitigation*; • **Software and application security** → Software security engineering; • **Machine learning** → Machine learning approaches;

KEYWORDS

Big data Analytics, Malware detection, Machine learning, Deep learning

ACM Reference Format:
Emmanuel Masabo, Kyanda Swaib Kaawaase, and Julianne Sansa-Otim. 2018. Big Data: Deep Learning for detecting Malware. In *SEIA '18: SEIA '18: Symposium on Software Engineering in Africa , May 27–28, 2018, Gothenburg, Sweden.* https://doi.org/10.1145/3195528.3195533

1 INTRODUCTION

Current malware detection systems have some limitations in detecting modern malware because these malware are polymorphic and spread quickly in large numbers. Existing systems are mainly signature based. Yuan et al.[23] reported that the detection rate of sophisticated zero-day and polymorphic malware by anti-virus products is between 25% and 50%. Signature-based systems use an already defined pattern to detect existing malware. They fail when a malware morphs itself programmatically while exhibiting the same functionalities. They should wait for a new pattern/signature to be created. This can't work because more malware are generated than signatures. Another thing is that a malware can change itself in an infinite number of instances. It's not possible to have an unlimited number of signatures and no database can store the unlimited amount of data.

This research discusses the art of applying big data analytics techniques in the effort of addressing the above-mentioned concerns. The question is why big data? Big data have a lot of advantages. They provide strong analytics and detailed insights of the problem. They are good at analyzing big datasets. As malware corpus keeps increasing constantly, big data can fit well in addressing this issue. Big data can help explore data in the real-time and take adequate decisions. Big data enhance machine learning models' prediction performances. We adopted deep learning(DL) models as they are very useful in analyzing big datasets and provide high accuracy [3, 23]. Figure 1 discusses the difference between traditional and big data approaches. The main highlight is that big data can work better in a real-time setting without necessarily having to create a repository of all data.

The main contributions of this paper are as follows:

(1) We propose a scalable big data based analytical approach for efficient malware detect using deep learning.
(2) We evaluate the robustness of our approach on a complex dataset and assess the outcome of performance metrics.

The rest of this paper is organized as follows. Section 2 provides a litterature on the impact of big data in malware detection and discusses the related existing detection techniques. The research methodology is provided in section 3. Results and discussion are provided in section 4. Conclusion and further work are presented in section 5.

2 RELATED WORK

2.1 Big data analytics

Big data is just a term used to describe large datasets that are complex and voluminous. They are either unstructured, semi-structured or structured [9, 10, 20]. They need advanced techniques to deal with them [17] as traditional data management software can't handle them. They can be analyzed to get insights that help in making better decisions [11]. Big data are characterized by 7V's (Volume, Velocity, Variety, Variability, Veracity, Visualization, Value)[6].

Volume is about how large is data. It can go from Gigabytes, Terabytes, Zettabytes or Yottabytes. Malware data are generated in high volumes as described in the above section.

Velocity is the speed of accessing data. As malware evolve and spread so fast, big data can help in counterfeiting their progress through proactive analysis and detection.

SEIA '18, May 27–28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5719-7/18/05…$15.00
https://doi.org/10.1145/3195528.3195533

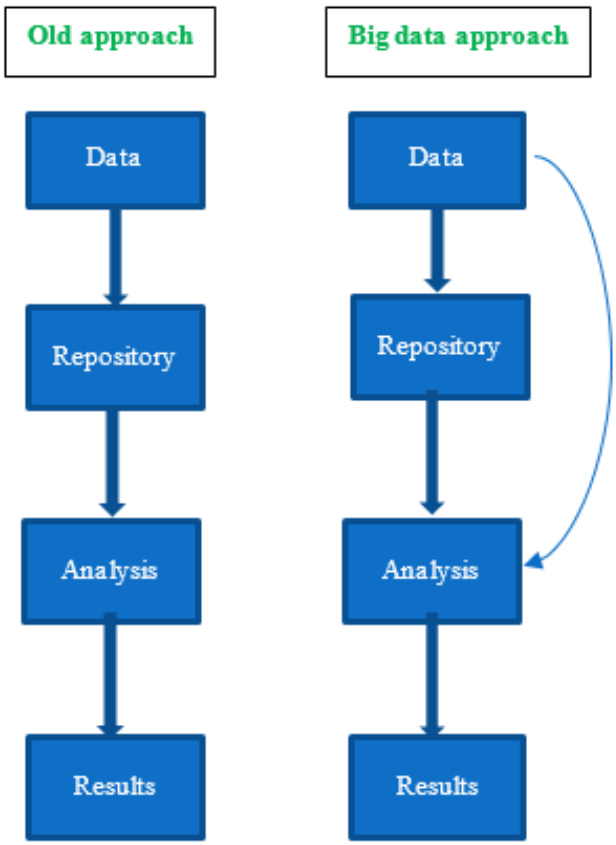

Figure 1: Traditional approach vs big data approach.

Variety is about different types of data and unstructured nature of the data. Malware are of different types and purposes. Big data are able to deal with these varieties efficiently.

Variability stipulates that the same data can have different meanings. Files can be either clean, malware or unwanted. Big data will certainly know how to analyze and identify them with a high accuracy.

Veracity is about the accuracy or quality of data. Data frequently comes full of noise. Big data techniques can deal with noise in order not to negatively impact the performance of detection models.

Visualization consists of graphics, charts and other plots that help in understanding the meaning of data and retrieving more details. Early exploratory analysis through visualization can give hidden details on the malware functionalities

Value is about how data can be processed to produce a valuable outcome. The most valuable outcome in malware analysis through big data is the models that detect with high accuracy.

Big data can help in detecting malware and identifying suspicious threats in the real time. In addition to that, big data is capable of providing a quick intelligence to launch an immediate automated response to an attack[2, 4, 7, 19]. Data to be analyzed can be fed from existing large repositories or can be streamed lively from Internet, social media or CCTV. Big data analytics technique can be used to perform a real-time analysis and provide adequate results.

2.2 Detection techniques

Meng at al. in [15] proposed a malware classification model called MCSMGS which combines deep learning with malware static genes based on API call sequences. Features were extracted as gene sequences. A convolutional network was constructed for analysis. Their experiments achieved an accuracy of 98% on a dataset composed of PE files downloaded from VX heaven repository [22].

Yuan at al. in [23] proposed a multistage analysis technique that uses deep learning to classify or detect malware. Their experiment gave an accuracy of 94.83% and 94.66% F1 score. They stipulated that Deep learning can perform well even though it runs slowly.

Bojan at al. in [3] proposed a deep learning method for classifying malware through system call sequences. They constructed a convolutional neural network as a means to extract best features for classification. n-gram features were extracted. Their model achieved 85.6% on precision and 89.4% on recall.

Kolosnjaji at al. in [12] developed an adaptive semi-supervised malware classification technique that combines advantages of static and dynamic analyses to get better results. Samples used were collected from Virus Share, maltrieve [13] and other private repositories. Execution sequences were collected using Cuckoo sandbox and other features were obtained using PEInfo and Yara. Their developed semi-supervised model achieved an accuracy of 97.5%. It outsmarted supervised learning model which achieved 96.9%.

Rhode at al. in [17] created a recurrent neural network-based approach, that was used to detect if a file was malicious of benign by just utilizing a small snapshot of behavioral data, instead of considering the post-execution log report. Their dataset was composed of 2345 malicious files from Virus Total, 2286 benign files from windows and 2876 ransomware from Virus Share. Their approach could achieve an accuracy of 94% within the first five seconds of execution.

Shibahara at al. in [18] proposed a dynamic malware analysis approach based on deep learning. The purpose of this approach was to increase the efficiency of dynamic analysis by determining when to stop it depending on the inconsistent change of malware communication purpose. Their experiments reduced the analysis time by 67.1% while keeping the overall coverage of collected malware to 97.9%. The dataset was composed of 29.562 malware samples.

Huang at al. in [8] proposed a Multi-Task Neural Network for Dynamic Malware Classification approach named MtNet. They used deep learning to classify benign from malware files. They trained the model on 4.5 million samples and achieved a classification error rate of 0.358% and low false positives under 0.07%.

3 METHODOLOGY

In order to solve our research problem of creating an analytical big data based approach for effective malware detection, we followed the following steps: *Dataset collection, Exploratory analysis, Preprocessing, Feature transformation, Model development*, and *Model evaluation.*

3.1 Dataset collection and preliminary tasks

The dataset used in this research was downloaded from a public repository github and provided by Marco Ramilli [16]. The dataset has multiple features extracted from a wide variety of executable

malware. These features were extracted using sandboxes tools. Every feature represents a specific caracteristic or a behaviour of a malware. Static and dynamic features were extracted. Static features represent some inner physical structures of a malware. Dynamic features represent the specific behaviours of a malware and can only be obtained at run time when a malware is executed. The extracted features covered a wide range of malware operations such as: *file activity*(open, read, delete, modify, move), *Registry activity*(open, create, delete, modify, move, query, close), *Service activity*(open, start, create, delete, modify), *Mutex* (create, delete), *Processes*(start, terminate), *Runtime DLLs*, *Network activity*(TCP, UDP, DNS, HTTP), *Hooking activity*, *Anti-analysis behaviors*, *Self-hiding behaviours*, etc.

Preliminary tasks included the creation of a full dataset from individual JSON files as downloaded from the repository. Another task was to clean those samples which had a negligible occurrence while considering the ones with higher occurrence percentage. We did this in order to keep a balanced dataset that could yield better results. Feature values were represented by the first byte of hashed strings. This was considered to be the evidence of a particular behavior. A single behavior could have many pieces of evidence during execution of the malware. Each evidence was recorded and hashed. In dealing with these hash values, we computed the number of specific unique pieces of evidences per feature per sample and populated the dataset with new values. The dataset is initially composed of 292 APT, 2020 Crypto, 431 Locker, 2019 Zeus and 1270 shadow brokers.

3.2 Exploratory analysis

By using the big data analytics visualization technique, we realized that the occurrence rate and the distribution of features are very different among various malware types. This is shown in figure 2. For instance, it can be seen that shadowbroackers are described by very few features, while crypto are described by almost all the available features in the dataset. Another observation is that some features are more important than others in classifying the malware as shown in Figure 3.

3.3 Preprocessing

In our experiments, we considered a binary classification problem where only two outcomes are expected. In the case of malware, this is commonly about determining if a program is either a malware or benign. As the dataset we have didn't provide benign samples, we decided to experiment with the most two occurring malware families. These are Crypto and Zeus as shown in figure 4. We assume that if a deep learning detection model can successfully learn through these binary settings of the dataset and get good accuracy; it can as well do the same when there are more than two categories to predict. This will be part of the future work where we will consider a multi-class setting problem.

The final experimental dataset was composed of 2957, out of which Crypto were 1815 and zeus were 1142. The number of remaining features was 89.

3.4 Feature transformation

The purpose of this process was to have good training data, well optimized to fit the requirements of machine learning algorithms. Feature transformation helps the learning process and provides an additional background experience to input data, thus enabling the learning algorithm to benefit from such experience [1]. During this process, new features mighty be created by either applying a derivation of some existing features or by exploiting feature interaction relationships. In addition to that, more transformation is needed on categorical as well as numerical features.

Our dataset was composed of numerical variables with very large differences among them. Transformation was needed to get good input features that will make learning algorithms more successful. We applied the standard scaler transformation. This method transforms data to a normal distribution within each feature and scales them such that the distribution is centered around a mean of 0 and standard deviation of 1. This format is suitable for most algorithms [5]. The feature is scaled based on the following equation:

$$\frac{x_i - \mu}{\sigma}$$

, with the mean calculated as:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and the standard deviation calculated as:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

, where x_i represents features in the dataset.

Categorical class labels(Crypto and zeus) were converted to numerical values using label encoding technique; thus having Crypto and Zeus represented as 0 and 1, respectively.

3.5 Model development

We adopted deep learning to develop our model. It was a neural network with 3 layers (one input layer, one hidden layer, and one output layer). The snapshot configuration of the model is shown in figure 5.

In order to assess the robustness of deep learning, we used Support Vector Machine (SVM). SVM is one of the powerful and most popular algorithms. It performs well in binary classification problems as well as multiple classification problems.

3.6 Evaluation

A good model should perform well on unseen data based on the knowledge gained from the training data. During our experiments, we only had a training dataset. In order to estimate the prediction accuracy of our model, we, therefore used a resampling technique called train_test split [14, 21], whereby a specific percentage of the data is held out for testing purposes and the remaining is used for training. In our case, we chose 80% for training and 20% for testing. We further extended our evaluation to assess the robustness of the model using performance metrics. The following metrics have been used.

Confusion matrix: it is used to describe the performance of a classifier. Outputs are presented in a table layout shape. Information given by the matrix is as follows: True Positive(TP), which

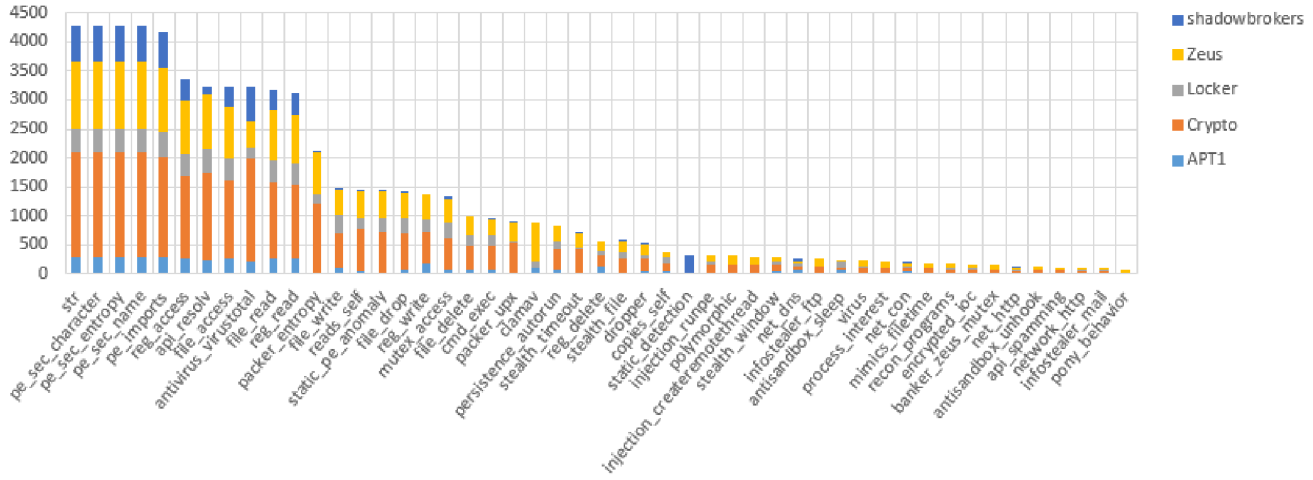

Figure 2: Distribution and occurrence rates of features in various malware categories.

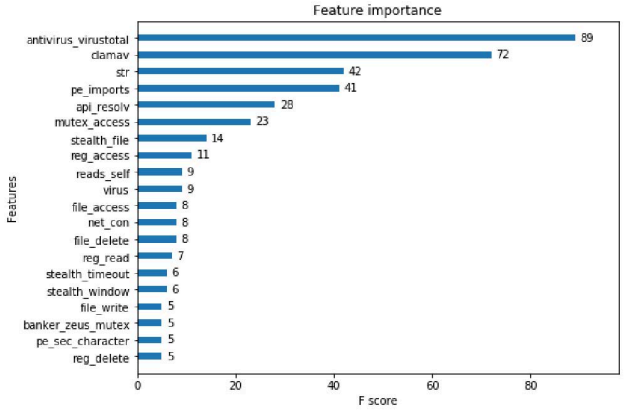

Figure 3: Feature importances

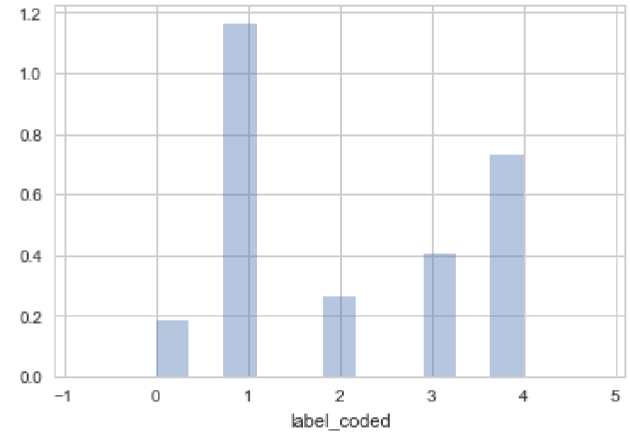

Figure 4: Distribution of classes in our sample. Classes are coded as: 'APT':0, 'Crypto':1, 'Locker':2, 'shadowbrokers':3, 'Zeus':4.

represents the number of malware correctly identified as being of a specific class 0. True Negative(TN) is the number of malware that are correctly identified as not being of class 0. False Positive (FP) represents the number of malware that are wrongly classified as being of class 0. False Negative (FN) shows the number of malware wrongly classified as not being of class 0.

Precision: evaluates how often the model is correct when it correctly detects malware as being of class 0. It is defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

Recall: evaluates how often the model can correctly detect malware as being of class 0. It is also called sensitivity. It is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 89)	7921
dense_20 (Dense)	(None, 8)	720
dense_21 (Dense)	(None, 1)	9

Total params: 8,650
Trainable params: 8,650
Non-trainable params: 0

Figure 5: Keras classifier model description.

F_measure: It is also known as the F1 score. This is a harmonic mean of recall and precision. It gives describes the robustness and precision of the model. It is defined as follows:

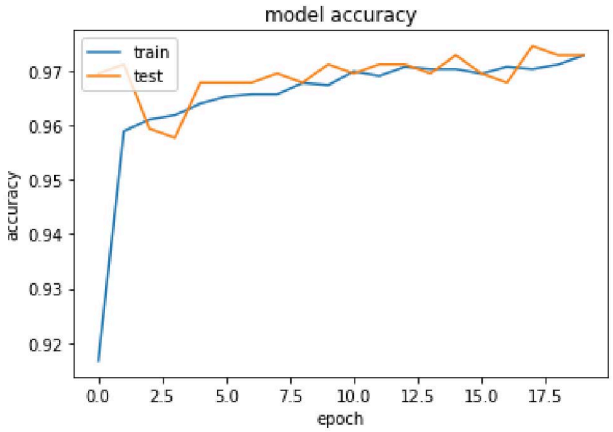

Figure 6: Model accuracy during training iterations.

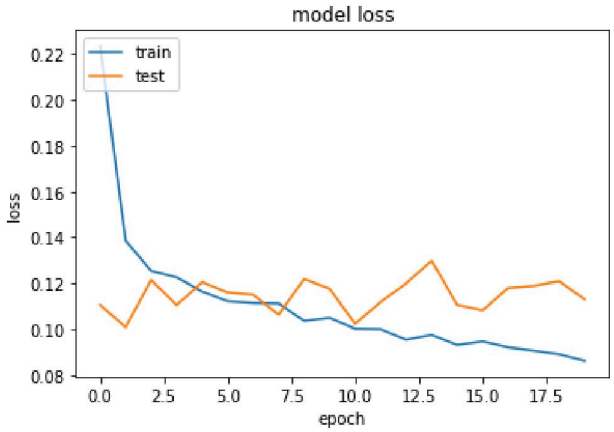

Figure 7: Model loss during training iterations.

$$F\_measure = 2 * \left(\frac{Precision * Precision}{Precision + Precision} \right)$$

ROC Curve: This summarizes graphically the performance of the model on all given classes.

4 EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Results

The dataset was split into train and test sets. Features were transformed to meet the requirements of chosen algorithms. A python deep learning library called Keras was used to build and implement the model. Model performances during training iterations are shown in figure 6 and figure 7 respectively. The model achieved the highest accuracy of 0.97 on both train and test sets. The model also achieved the lowest log loss of 0.0928. Other results are reported in table 1. It can be seen that Keras deep learning has performed better than SVM. The details are shown in figure 8. Figure 9 shows the confusion matrix performances.

The detection performance of our models are described by the areas under the Receiver Operating Curves (ROC). A ROC curve is generated by plotting the true positive rate (TPR) against the false positive rate (FPR) at differing threshold settings. The area under the ROC curve (AUC) is supposed to be 1 for a perfect model. The closer AUC is to 1, the better the model. Figure 10 and figure 11 show the ROC curves for our malware classes (Crypto: 0 and Zeus: 1) as well as the computed AUC for Deep learning and SVM models respectively. In both figures, AUC are above 90% which is a good indicator of a high detection performance. Deep learning has 99% ability to detect of Crypto and Zeus malware respectively. SVM has 98% ability to detect Crypto and Zeus malware respectively.

4.2 Discussion

The model developed is capable of detecting malware with promising accuracy. The robustness of this model is explained by the good performances achieved on different evaluation metrics. However, during the training process, the model was slow. This can be taken into consideration for future work. Another future plan will be to also increase the accuracy.

Table 1: Evaluation results

Evaluation Metric	SVM	Keras Deep Learning
Precision	0.94893617	0.963562753
Recall	0.91393443	0.975409836
F1	0.93110647	0.969450102
Accuracy on Training Set	0.95	0.97
Accuracy on Test Set	0.94	0.97

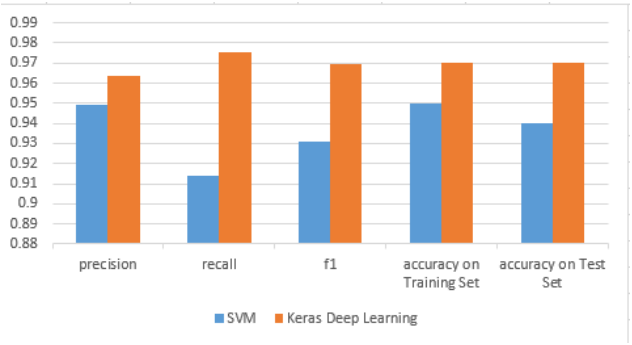

Figure 8: Comparing the SVM and Deep learning performances.

Comparing our model to MCSMGS [15] as shown in table 2, we can say that our model is robust even though the accuracies differ slightly. On testing configuration, the test set used by MCSMGS is very small. MCSMGS only used static features and couldn't capture the inner functionalities of malware. In our case, we used static and dynamic features which identify key functionalities of the malware. The F1 score of MCSMGS is not provided. Our F1 score is very promising and reached 99%.

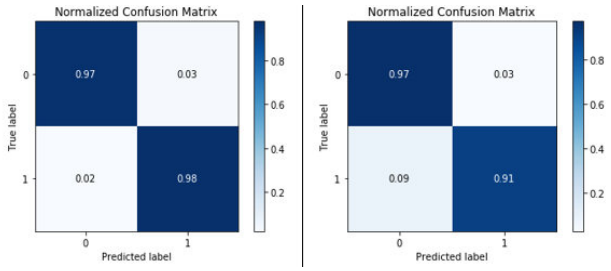

Figure 9: Comparision of TPR and FPR between Deep Learning shown on the left and SVM shown on the right side.

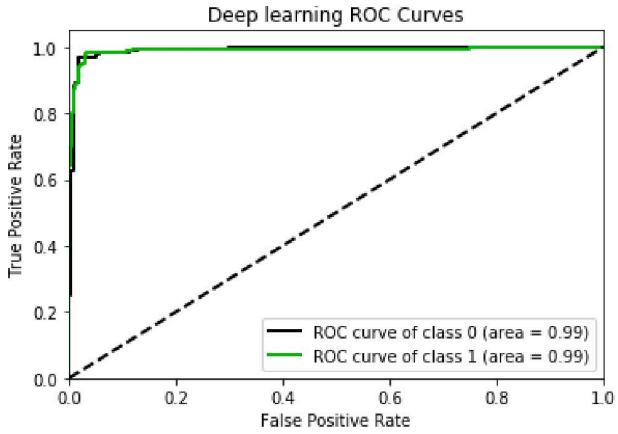

Figure 10: SVM predictive ability of Crypto (clas 0) and Zeus(class 1) as represented by Areas under ROC curve (AUC).

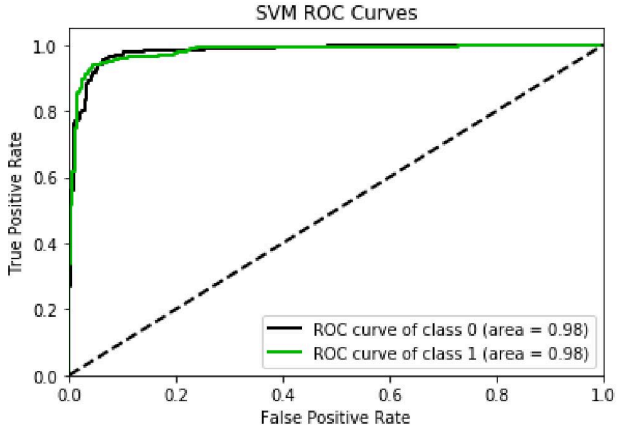

Figure 11: SVM predictive ability of Crypto (clas 0) and Zeus(class 1) as represented by Areas under ROC curve (AUC).

Table 2: Comparison with previous work

	Static Features	Dynamic Features	Accuracy	F1 score	Training set	Test set
MCSMGS	Yes	No	98%	-	90%	10%
Our model	Yes	Yes	97%	99%	80%	20%

5 CONCLUSION

In this paper, we proposed a malware detection approach based on big data analytics and machine learning. We used key features that represent the deeper intrinsic malware structure and behavioral. To build the model, we adopted Deep Learning and Support Vector Machine. Experiments show that deep learning achieved a better accuracy of 97% compared to 95% achieved by SVM. In the future, we aim to improve the speed of the model and explore further tuning settings to make our model more robust and more accurate. These improvements should consider not only binary classification problems such as (malware vs malware) or (malware vs benign); but also should handle multi-class setting situations where we have more than two categories of either malware/malware or malware/benign groups of samples.

ACKNOWLEDGMENTS

We would like to thank the Mobility to Enhance Training of Engineering Graduates in Africa (METEGA) and the Regional Universities Forum for Capacity Building in Agriculture (RUFORUM) for supporting our research.

REFERENCES

[1] Amazon. 2015. *Amazon Machine Learning Developer Guide.* Technical Report. https://docs.aws.amazon.com/machine-learning/latest/dg/machinelearning-dg.pdf
[2] Brian Bloom. 2012. Big data analytics defining new malware strategy. (2012). https://www.itworldcanada.com/article/big-data-analytics-defining-new-malware-strategy/45468
[3] Kolosnjaji Bojan, Zarras Apostolis, Webster George, and Eckert Claudia. 2016. Deep Learning for Classification of Malware System Call Sequences. *AI 2016: Advances in Artificial Intelligence* (2016).
[4] Tony Bradley. 2012. FireAMP Fights Malware with Big Data Analytics. (2012).
[5] Jason Brownlee. 2016. How To Prepare Your Data For Machine Learning in Python with Scikit-Learn. (2016). https://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/
[6] Ashley DeVan. 2016. The 7 V's of Big Data. (2016). https://www.impactradius.com/blog/7-vs-big-data/
[7] Cath Everett. 2015. Big data - The future of cyber-security or its latest threat? *Computer Fraud and Security* 2015, 9 (2015), 14–17. https://doi.org/10.1016/S1361-3723(15)30085-3
[8] Wenyi Huang and Jack W. Stokes. 2016. MtNet: A multi-task neural network for dynamic malware classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9721 (2016), 399–418. https://doi.org/10.1007/978-3-319-40667-1_20
[9] IBM. 2013. Smarter security intelligence. (2013), 1–8.
[10] Ibm.com. [n. d.]. Big Data Analytics | IBM Analytics. ([n. d.]). https://www.ibm.com/analytics/hadoop/big-data-analytics
[11] SAS Insights. [n. d.]. What is Big Data and why it matters. ([n. d.]). https://www.sas.com/en{_}us/insights/big-data/what-is-big-data.html
[12] Bojan Kolosnjaji, Apostolis Zarras, Tamas Lengyel, George Webster, and Claudia Eckert. 2016. Adaptive Semantics-Aware Malware Classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 419—-439.
[13] Krmaxwell. 2015. A tool to retrieve malware directly from the source for security researchers. (2015). https://github.com/krmaxwell/maltrieve
[14] Arvind Mahindru and Paramvir Singh. 2017. Dynamic Permissions based Android Malware Detection using Machine Learning Techniques. *Proceedings of the 10th Innovations in Software Engineering Conference on - ISEC '17* (2017), 202–210. https://doi.org/10.1145/3021460.3021485
[15] Xi Meng, Zhen Shan, Fudong Liu, Bingling Zhao, Jin Han, Hongyan Wang, and Jing Wang. 2017. MCSMGS: Malware Classification Model Based on Deep

Learning. *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (2017), 272–275. https://doi.org/10.1109/CyberC.2017.21

[16] Marco Ramilli. 2016. Malware Training Sets: a machine learning dataset for everyone. (2016). http://marcoramilli.blogspot.it/2016/12/malware-training-sets-machine-learning.html

[17] Matilda Rhode, Pete Burnap, and Kevin Jones. 2017. Early Stage Malware Prediction Using Recurrent Neural Networks. December (2017), 1–28. arXiv:1708.03513 http://arxiv.org/abs/1708.03513

[18] Toshiki Shibahara, Takeshi Yagi, Mitsuaki Akiyama, Daiki Chiba, and Takeshi Yada. 2016. Efficient Dynamic Malware Analysis Based on Network Behavior Using Deep Learning. *2016 IEEE Global Communications Conference (GLOBECOM)* (2016), 1–7. https://doi.org/10.1109/GLOCOM.2016.7841778

[19] TechSpective. 2015. Big data analytics leads the way for next-gen malware protection. (2015). https://techspective.net/2015/04/27/big-data-analytics-leads-the-way-for-next-gen-malware-protection/

[20] Muhammad Habib ur Rehman, Chee Sun Liew, Assad Abbas, Prem Prakash Jayaraman, Teh Ying Wah, and Samee U. Khan. 2016. Big Data Reduction Methods: A Survey. *Data Science and Engineering* 1, 4 (2016), 265–284. https://doi.org/10.1007/s41019-016-0022-0

[21] Jake VanderPals. 2016. *Python Data Science Handbook | Python Data Science Handbook*. O'Reilly, Sebastopol, CA. 541 pages. https://jakevdp.github.io/PythonDataScienceHandbook/index.html

[22] VX-Collection. 2016. VX Heaven windows virus collection. (2016). https://archive.org/details/vxheaven-windows-virus-collection

[23] Xiaoyong Yuan. 2017. PhD Forum: Deep Learning-Based Real-Time Malware Detection with Multi-Stage Analysis. *2017 IEEE International Conference on Smart Computing, SMARTCOMP 2017* (2017), 1–2. https://doi.org/10.1109/SMARTCOMP.2017.7946997