Bidirectional Long Short-Term Memory classifier assist for intelligent ransomware detection in Android OS

Jakub Siłka[1]

[1]Faculty of Applied Mathematics, Silesian University of Technology Kaszubska 23, 44-100 Gliwice, Poland

Abstract
A precise description of an invented, complementary method for android's system security is presented in this study. I have created such a system in order to provide casual smartphone users with a useful tool to prevent the ransomware type attacks. With the purpose of obtaining a satisfying outcome, I used BiLSTM and a optimizing algorithm NAdam. I managed to receive a system's accuracy up to over 93% therefore creating a prospectS of complementing the existing security system especially when it is not clear whether a certain application program is safe.

Keywords
Ransomware, Cybersecurity, Deep Learning, Bidirectional LSTM, Artificial Intelligence

1. Introduction

No personage can deny, that nowadays, we live in the most digitization-based times. Therefore, a lot of different methods are increasingly being developed to protect ourselves in all sorts of new ways such as Cabaj, Gregorczyk, and Mazurczyk [1] or Hu and Tan [2]. Although we do continuously find a new way to use it on our behalf, simultaneously some threats associated with the violation of personal devices are also occurring more often. Therefore, more and more often there are very ingenious ways to counteract these threats, such as Venkatraman, Alazab, and Vinayakumar [3], [4] and Vinayakumar, Soman, Senthil Velan, and Ganorkar [5]. Actually, most of the youth keeps a constant contact with internet via smartphone, rather than desktop computer. According to the fact, that an android figures prominently in a mobile operating systems market and a fact that in the last years we've observed increased number of threats related to the android I do think that the already existing security methods have to be revised this will accelerate the progress of many scientific discoveries such as Ferrante, Malek, Martinelli, Mercaldo, and Milosevic [6]. In addition, along with the companies tendency to make wearing a device as intuitive and easy as possible, more people are apt to use it - especially elders and very young ones. However, i should mentioned, that such cyber-attacks can be tremendously reduced just by applying an appropriate hygiene of using an device - including not downloading the data from suspected pages etc. However, in order to detect the greatest threats, some method

IVUS'21: 26th International Conference Information Society and University Studies, April 23, 2021, Kaunas, Lithuania
kubasilka@gmail.com (J. Siłka)
000-0001-9133-4388 (J. Siłka)

of machine learning is needed if the contractor depends on the flexibility of the system itself, they are very well known as it can be observed in many publications like YIL [7] and Pouyanfar, Sadiq, Yan, Tian, Tao, Reyes, Shyu, Chen, and Iyengar [8]. However, in order to enable the network to gain any significant advantage, there was a need for the network to remember, so it is worth remembering about works such as Cao, Yang, Tang, and Lu [9], Sherstinsky [10] or Yu, Si, Hu, and Zhang [11]. In this work a model of Recurrent Neural Network was applied as classification of ransom software. Similar idea of those architectures was applied in i previous research for body pose prediction Woźniak, Wieczorek, Siłka, and Połap [12] and malware threads detection Woźniak, Siłka, Wieczorek, and Alrashoud [13]. The idea used in this paper is oriented on detection of potential dangerous situations in networking solutions. Proposed detection model was developed by using LSTM based model of neural network architectures. Numerical vectors are analyzed by BiLSTM and final classification is returned to the system. In this method various training algorithms were used, and during research the best trained model was selected. Research results show high efficiency of this proposition.

2. System

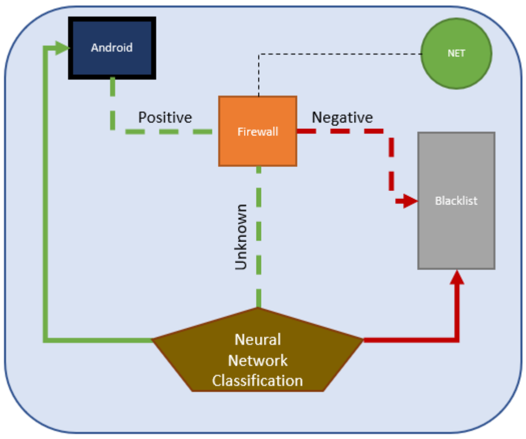

Figure 1: Presentation of the system operation diagram.

As one can observe in the fig.1, the entire system was created in order to supplement the already existing backup systems, such as firewall. The latter supervise the data stream between the internet and a device. That is why the system enables data to flow solely under one condition - if these aren't recognised by the program as explicitly harmful. However, if the network classifies a given vector as ransomware, it adds information about it to the blacklist of a certain phone. As an outcome, if an identical vector would be classified again by the backup system in the future - it will be possible to immediately determine how could it been classified by the other network. Hence, it is a method that solely revises a user's secure. In the case of such model existence, presumably the blacklist would be send periodically to the labs, in which the real classification would be determined.

3. Dataset

The data were gathered and downloaded from the source Lashkari, Kadir, Taheri, and Ghorbani [14]. They contain a great amount of a particular abstract class - in this case however, I chose the beningn's and ransomware's classes. I underpin this decision with the prospect of providing a refined security for a casual user. All of the classes use many attributes, such as:

1. Total Length of Fwd Packets
2. Total Length of Bwd Packets
3. Fwd Packet Length Max
4. Fwd IAT Total
5. Fwd Packet Length Mean
6. Fwd IAT Max
7. Bwd Packet Length Max
8. Bwd Packet Length Min
9. Fwd Avg Bulk Rate
10. Idle Min
11. Flow Bytes/s
12. Flow Packets/s,

4. Applied mathematical solutions to the problem.

In order to appropriately train this Neural Network I had to establish which optimising algorithm would be the most convenient. Following a lot of initial trails, I settled that the NAdam algorithm was the best - that is, practically speaking, Adam algorithm with the application of Nesterov momentum.

NAdam formula is:

$$m_t = h_1 m_{t-1} + (1 - h_1) g_t, \tag{1}$$

$$v_t = h_2 v_{t-1} + (1 - h_2) g_t^2, \tag{2}$$

where h is parameter of hyper-parameter and g is gradient value of the error function. Values $\hat{m_t}$ and $\hat{v}_t$ are calculating according to equations:

$$\hat{m_t} = (1 - h_1) g_t + h_{1t+1} m_t \tag{3}$$

$$\hat{v}_t = \frac{v_t}{1 - h_2^t}. \tag{4}$$

Using equations above, the weights in ANN architecture was defined as follows:

$$w_{t+1} = w_t - \eta \frac{\hat{m_{t+1}}}{\sqrt{h_{2t+1}} + \epsilon}, \tag{5}$$

where η is a learning rate (in this case 0.0005) andϵ is a constant value.

We applied NEG to Adam as:

$$w_t = w_{t-1} - \eta \frac{h_1 m_{t-1}}{h_2 v_{t-1} + (1-h_2) g_t^2 + \epsilon} - \eta \frac{(1-h_1) g_t}{\sqrt{h_2 n_{t-1} + (1-h_2) g_t^2 + \epsilon}} \tag{6}$$

5. Experiment and Results

The data have been downloaded from the Lashkari, Kadir, Taheri, and Ghorbani [14]. The authors had carefully worked with the model "CCS to caputure", depicting 200K applications. The VirusTotal was used in order to mark particular abstraction's classes. In a study, I applied the Ransomware and benign abstraction's classes. Following the process of the data' preparation - which simply refers to make it possesing two abstraction's classes "[Ransomware, Benign]", the NN's architecture building process was started off. In order to attain satisfying results, I have scrutinized different NN models and also compared a great scope of known approaches - starting with casual ANN, going through RNN and finally getting up to BiLSTM, which actually proved to be the most suitable, available model. However, one should notice other optimising algorithms such as AdaGrad, Adam, and NAdam, that I used. The latter emerged as the most efficient. After established the best combination, I had to determine the most accurate network architecture.

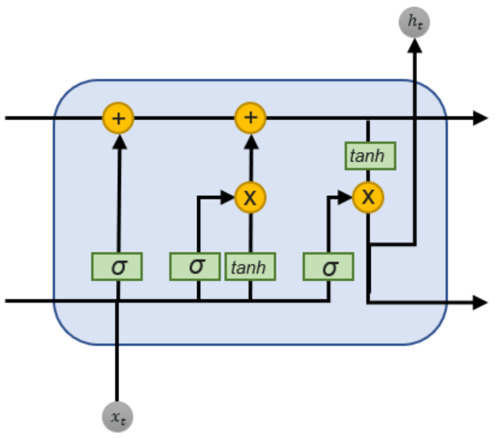

Figure 2: Above is the Long Short Term Memory gate which is used in the architecture of the entire system

As I managed to make this system remember the results, I had to wise up why didn't the results increase along with the time. After an in-depth analysis of performed experiments and received outcomes it was settled that it is necessary to enlarge the communication in the particular layers as well as in between them. That is why I have chosen the LSTM Bidirectional model. As I created the LSTM's Gate fig.2 and the specific layer, I proceeded to form an entire network's architecture, essential for classifying the set. It is crucial to understand that the BiLSTM is more sensitive to changes in the values of variables architecture than casual ANN

Table 1
Most common machine learning metrics for testing different optimization algorithms in short term testing on low number epoch

Algorithm	Acc.	Prec.	Rec.	F1	Spec.	FDR	FPR	FNR	FOR	NPV
NAdam	78.5%	78.5%	78.6%	1.5	76.0%	77.6%	24.0%	18.9%	20.4%	79.5%
ADAM	76.9%	76.8%	76.7%	1.5	76.2%	79.4%	23.7%	22.5%	25.9%	74%
Adamax	67.6%	67.5%	67.5%	1.3	65.5%	69.1%	34.4%	30.4%	34%	65.9%
Adagrad	54.3%	55.1%	54.9%	1.1	51.2%	45.1%	48.7%	41%	35.2%	64.7%
Adadelta	53.6%	60%	50.9%	1.1	66.6%	98.5%	33.3%	46.6%	96.7%	3.2%
Ftrl	52.9%	26.4%	50.0%	0.6	0.0%	100.0%	0.0%	47.1%	100.0%	0.0%
SGD	52.9%	26.4%	50.0%	0.6	0.0%	100.0%	0.0%	47.1%	100.0%	0.0%

which entails the requirement of huge amount of time-consuming tests. Following the latter, I resurfaced the most accurate model. The configuration that yielded the best result is displayed above fig.4. The sequence started off with an use of BiLSTM layer consists of 1000 neurons - then the Dropout operation was conducted up to 60 percent. fig.4 Subsequently, the entire data underwent the next layer that consisted 500 neurons and consecutively, a dropout (whereas the values were the same as in the first dropout) along with another layer, although now, with 120 neurons. After that, the data moved towards the last, hidden layer of 60 neurons. All of those layers were linked with each other by the Dropout operation with the value 0.3. In order to get all the results within such network, the last synapses underwent the last Dropout operation with the values equal to $\frac{1}{5}$.

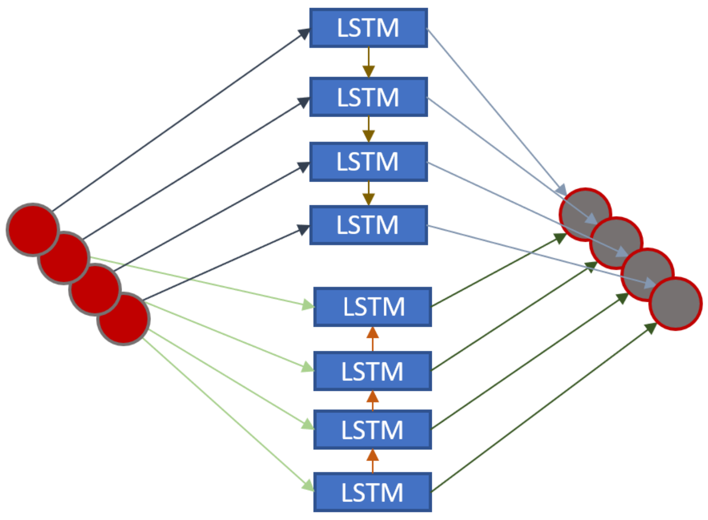

Figure 3: Diagram of the BiLSTM network, the data goes to the neurons, red circles with a gray rim to hit the two LSTM layers that are interconnected only to then return to normal.

The graphs generated as well as the confusion matrix for two of best algorithms in short-term

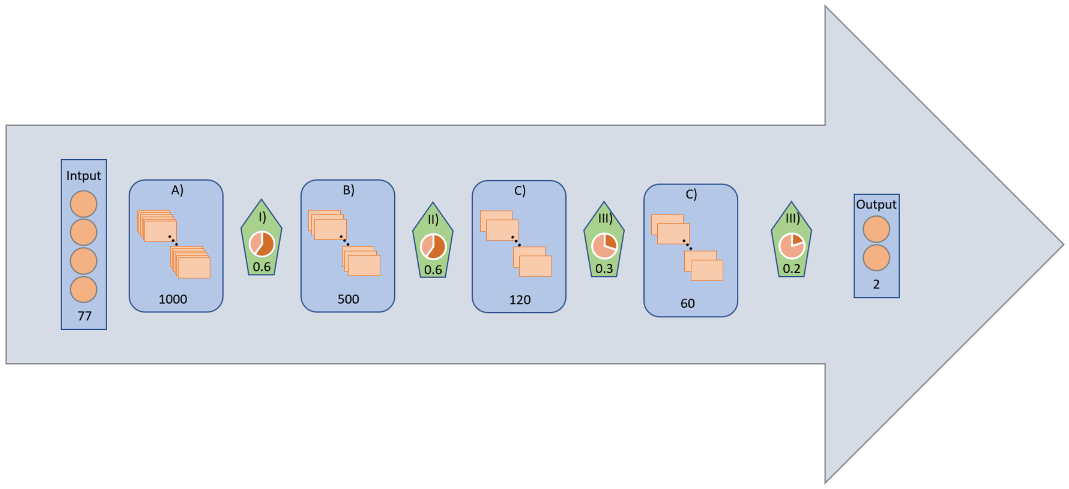

Figure 4: The architecture of the neural network that achieved the highest results in the research.

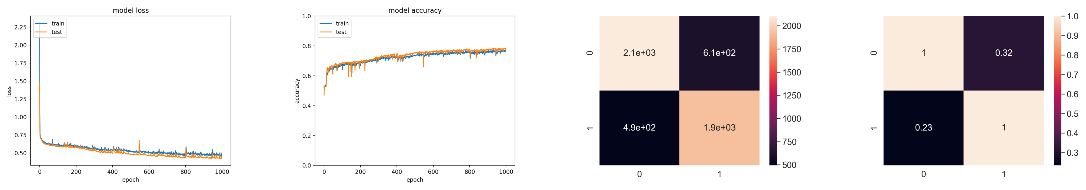

Figure 5: Plots and confusion matrix for NAdam optimization

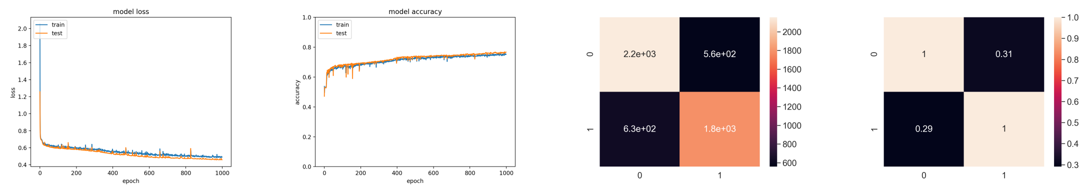

Figure 6: Plots and confusion matrix for ADAM optimization

learning are presented below.

As it is shown in the table above, it turned out that NAdam had the best performance and thus, this algorithm was selected for the following long-term tests. The long duration of a training is due to a way bigger complexity of computational whilst using BiLSTM - in contrast to casual ANN network. After a long training, using suitable architecture, I managed to achieve accuracy equal to 93.35% in the process of detecting dangerous programs.

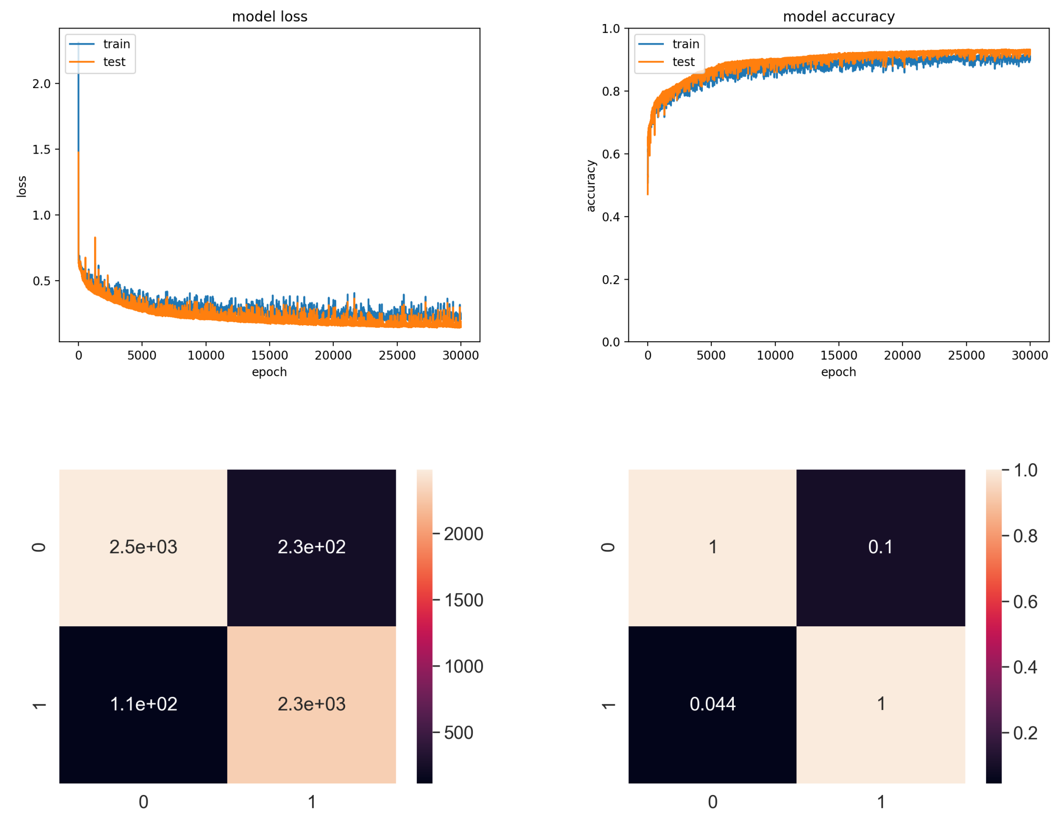

Figure 7: Plots and confusion matrix for the best configuration of each component in long-term neural training.

6. Conclusion

In the above study, i demonstrated a method to detect various approaches of protecting the android system against threats associated with ransomware. Moreover, I have compared those different approaches - and indicated which of the optimising algorithm or architecture was the most suitable and efficient. Following an aforementioned scrutinizing process of the obtained results, it turned out that the NAdam on BiLSTM architecture is the most accurate solution - along with this method I managed to get 93.35% accuracy. In this way, I adequately complemented the full scope of already exiting approaches created to prevent ransomware. In the foreseeable future, one ought to concentrate particularly on the process of augmentation the quantity of threats that can be detected with high accuracy.

Acknowledgments

Author would like to acknowledge contribution to this research from the Rector of the Silesian University of Technology, Gliwice, Poland, under program "Initiative of Excellence—Research University" grant no. 08/IDUB/2019/84.

References

[1] K. Cabaj, M. Gregorczyk, W. Mazurczyk, Software-defined networking-based crypto ransomware detection using http traffic characteristics, Computers Electrical Engineering 66 (2018) 353–368. URL: https://www.sciencedirect.com/science/article/pii/S0045790617333542. doi:https://doi.org/10.1016/j.compeleceng.2017.10.012.

[2] W. Hu, Y. Tan, Black-box attacks against rnn based malware detection algorithms, arXiv preprint arXiv:1705.08131 (2017).

[3] S. Venkatraman, M. Alazab, R. Vinayakumar, A hybrid deep learning image-based analysis for effective malware detection, Journal of Information Security and Applications 47 (2019) 377–389. URL: https://www.sciencedirect.com/science/article/pii/S2214212618304563. doi:https://doi.org/10.1016/j.jisa.2019.06.006.

[4] B. Alsulami, S. Mancoridis, Behavioral malware classification using convolutional recurrent neural networks, in: 2018 13th International Conference on Malicious and Unwanted Software (MALWARE), 2018, pp. 103–111. doi:10.1109/MALWARE.2018.8659358.

[5] R. Vinayakumar, K. P. Soman, K. K. Senthil Velan, S. Ganorkar, Evaluating shallow and deep networks for ransomware detection and classification, in: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 259–265. doi:10.1109/ICACCI.2017.8125850.

[6] A. Ferrante, M. Malek, F. Martinelli, F. Mercaldo, J. Milosevic, Extinguishing ransomware-a hybrid approach to android ransomware detection, in: International Symposium on Foundations and Practice of Security, Springer, 2017, pp. 242–258.

[7] A novel wavelet sequence based on deep bidirectional lstm network model for ecg signal classification, Computers in Biology and Medicine 96 (2018) 189–202. URL: https://www.sciencedirect.com/science/article/pii/S0010482518300738. doi:https://doi.org/10.1016/j.compbiomed.2018.03.016.

[8] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, S. Iyengar, A survey on deep learning: Algorithms, techniques, and applications, ACM Computing Surveys (CSUR) 51 (2018) 1–36.

[9] Y. Cao, F. Yang, Q. Tang, X. Lu, An attention enhanced bidirectional lstm for early forest fire smoke recognition, IEEE Access 7 (2019) 154732–154742. doi:10.1109/ACCESS.2019.2946712.

[10] A. Sherstinsky, Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, Physica D: Nonlinear Phenomena 404 (2020) 132306.

[11] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: Lstm cells and network architectures, Neural computation 31 (2019) 1235–1270.

[12] M. Woźniak, M. Wieczorek, J. Siłka, D. Połap, Body pose prediction based on motion sensor data and recurrent neural network, IEEE Transactions on Industrial Informatics 17 (2020) 2101–2111.

[13] M. Woźniak, J. Siłka, M. Wieczorek, M. Alrashoud, Recurrent neural network model for iot and networking malware threads detection, IEEE Transactions on Industrial Informatics (2020).

[14] A. H. Lashkari, A. F. A. Kadir, L. Taheri, A. A. Ghorbani, Toward developing a systematic approach to generate benchmark android malware datasets and classification, in: 2018 International Carnahan Conference on Security Technology (ICCST), IEEE, 2018, pp. 1–7.