Multi-Class classification of vulnerabilities in smart contracts using AWD-LSTM, with pre-trained encoder inspired from natural language processing

IOP SciNotes

ARTICLE

Multi-Class classification of vulnerabilities in smart contracts using AWD-LSTM, with pre-trained encoder inspired from natural language processing

Ajay K Gogineni[1,2], **S Swayamjyoti**[1,3], **Devadatta Sahoo**[1], **Kisor K Sahu**[2,3,4] and **Raj Kishore**[2,3]

[1] NetTantra Technologies (India) Pvt. Ltd, Bhubaneswar, Odisha 751021, India
[2] Virtual and Augmented Reality Centre of Excellence, IIT Bhubaneswar 752050, India
[3] School of Minerals, Metallurgical and Materials Engineering, IIT Bhubaneswar 752050, India
[4] Centre of Excellence for Novel Energy Materials, IIT Bhubaneswar 752050, India

E-mail: kisorsahu@iitbbs.ac.in

Keywords: machine learning, smart contracts, LSTM, AWD-LSTM, classification, invocation depth

Supplementary material for this article is available online

OPEN ACCESS

RECEIVED
14 September 2020

REVISED
17 November 2020

ACCEPTED FOR PUBLICATION
23 November 2020

PUBLISHED
3 December 2020

Abstract

Vulnerability detection and safety of smart contracts are of paramount importance because of their immutable nature. Symbolic tools like OYENTE and MAIAN are typically used for vulnerability prediction in smart contracts. As these tools are computationally expensive, they are typically used to detect vulnerabilities until some predefined invocation depth. These tools require more search time as the invocation depth increases. Since the use of smart contracts increases rapidly, their analysis becomes difficult using these traditional tools. Recently, a machine learning technique called Long Short Term Memory (LSTM) has been used to predict the vulnerability of a smart contract. In the present article, we present how to classify smart contracts into Suicidal, Prodigal, Greedy, or Normal categories using Average Stochastic Gradient Descent Weight-Dropped LSTM (AWD-LSTM), a variant of LSTM. We reduced the class imbalance by considering only distinct opcode combinations for normal contracts and achieved a weighted average F1 score of 90.0%. Such techniques can be utilized in real-time to analyze a large number of smart contracts and to improve their security.

1. Introduction

'Smart contract' (SC), a term coined by Nick Szabo in 1996 [1], is an extended idea of a blockchain. Smart contracts are digitally written set of transaction protocols which are automatically executed during transactions between mutually distrusted nodes without the mediation of any centralized trusted authority. Recently, many platforms have been used for writing smart contracts like Contract Oriented Language (COL), Ethereum, and Rootstock (RSK) [2, 3]. The contracts for Ethereum are written in the Ethereum Virtual Machine (EVM) and programmed through a language called Solidity [4]. As these smart contracts hold virtual coins worth thousands of USD, they are prone to vulnerabilities like bug implantation in contracts by attackers. There are few incidences of such exploitations of smart contracts. For example, in June 2016, a decentralized investment fund named DAO (Decentralized Autonomous Organization) lost approximately $70 million due to the stealing of over 3.6 million Ether [5]. In November 2017, a security alert was issued by Parity Technologies, saying that their parity wallet (multi-sig wallets) was affected due to which $300 million was frozen [6]. Presently, due to the availability of high computing capabilities, machine learning methods are now becoming a popular choice in analyzing data from different fields, such as analyzing bio-medical image scans [7, 8], Satellite images [9], materials characterization [10, 11], share market [12], etc. There are several examples of the adoption of machine learning tools for different types of security attack detection. Du *et al* [13] used a deep LSTM model for anomaly detection in systems. Similarly, Shen *et al* [14] used Recurrent Neural Network (RNN) having a sequence memory architecture for forecasting security events on a computer. Shin *et al* [15] used LSTM for the

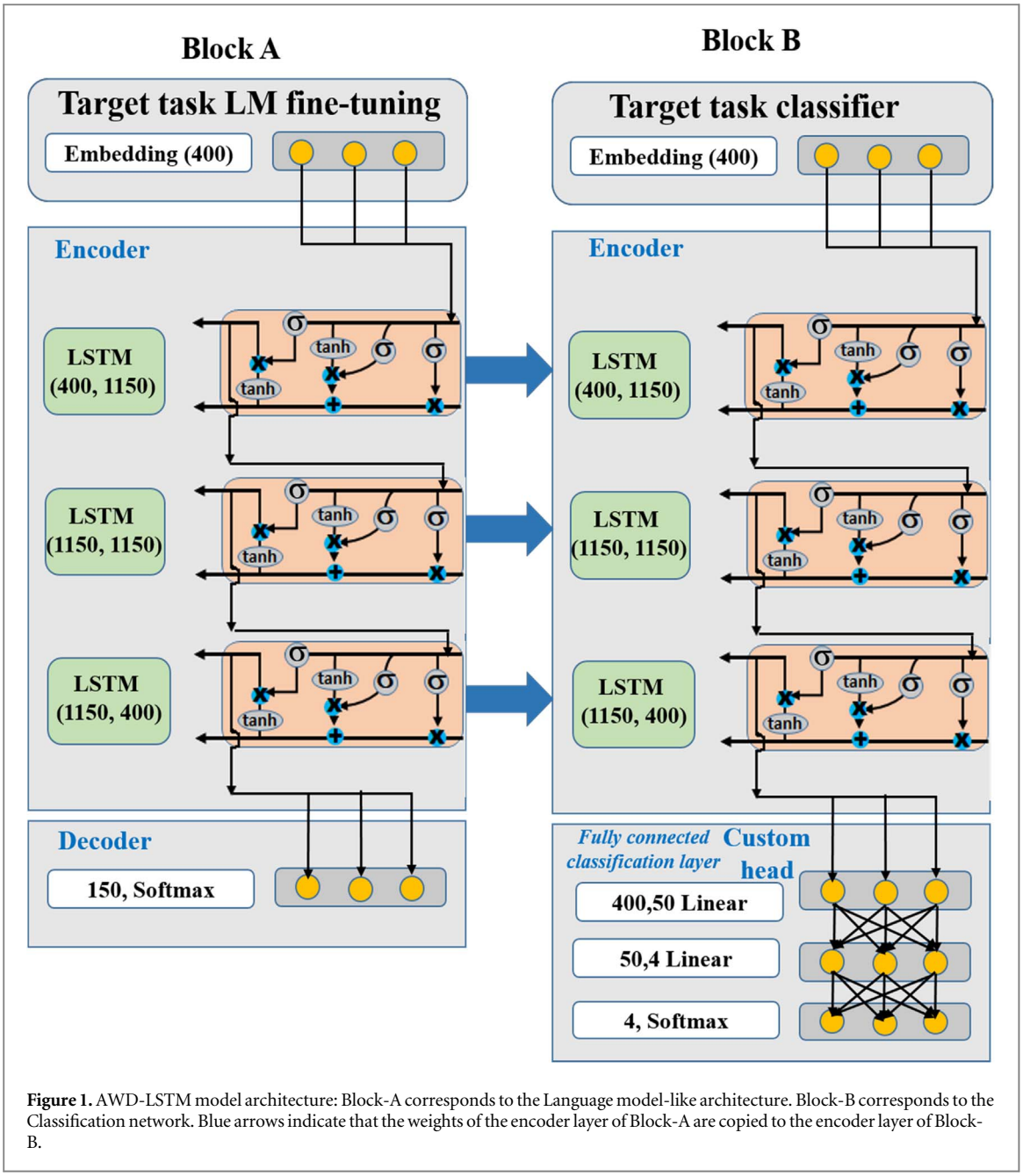

Figure 1. AWD-LSTM model architecture: Block-A corresponds to the Language model-like architecture. Block-B corresponds to the Classification network. Blue arrows indicate that the weights of the encoder layer of Block-A are copied to the encoder layer of Block-B.

identification of functions in binaries. These examples indicate that machine learning tools can be used to understand and improve the functioning of these cryptocurrency-based transactions. It will be interesting to further explore the utility of machine learning to predict the vulnerabilities, if present in the smart contracts. Some relevant key-concept generally used in this article, which are useful in understanding the intersection of machine learning and smart contract are briefly discussed in *supplementary S1* is available online at stacks.iop.org/IOPSN/1/035002/mmedia. The four different categories, (a) Suicidal, (b) Prodigal, (c) Greedy and, (d) normal SCs (discussed in *supplementary S1*.4) are classified using AWD-LSTM machine learning model (discussed in section 2.2). In the present model, we have combined a pre-trained encoder with the 'custom head' to increase the classification efficiency motivated by ULMFIT, which is used for the NLP application [16].

2. Methods

2.1. Data preparation

The SC data that is analyzed in this article is obtained from the work of Tann *et al* [17]. They sourced the original data from Google BigQuery and removed the false-positives present in it [18] and used MAIAN to obtain the labels based on vulnerability [19]. This pre-processed dataset has 892913 addresses, labelled in five different vulnerability categories as Type-1: Suicidal, Type-2: Prodigal, Type-3: Greedy, Type-4 normal SCs, and Type-5:

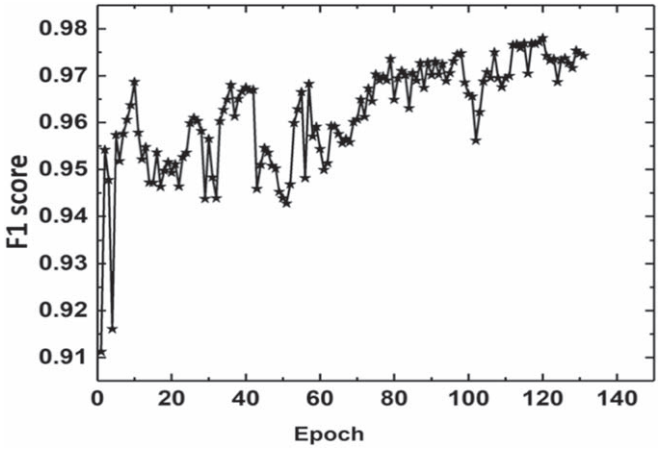

Figure 2. Variation of *F1* score on validation dataset with epochs.

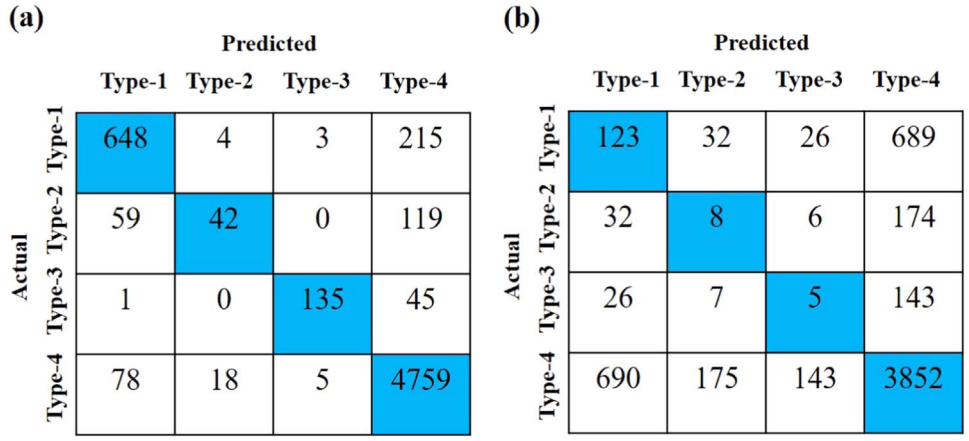

Figure 3. Confusion matrix for all categories for opcodes belonging to the test sets (a) using AWD-LSTM model and (b) using random guess.

Prodigal and Greedy both, with the number of SCs falling under them are 5801, 1461, 1207, 884273, and 171 respectively. Of these, we have selected the first four types. SCs of Type-5 were not considered as Type-5 is a composite category and has relatively fewer SCs which will create huge class imbalance. When an SC is invoked, a new address is appended, whereas the opcode combination remains unchanged so a single specific opcode combination will refer to multiple instances of addresses. We can reduce the computational efforts by considering only the distinct combinations of the opcodes truly representing distinct situations. While there were 892742 distinct addresses, only 34822 were distinct opcodes combinations. To reduce class imbalance, the opcode combinations for Types 1–3 were retained while for Type-4, only the unique combinations were retained. So we have analyzed 40,877 opcode combinations (Types 1–4 as 5801, 1461, 1207, and 32408, respectively).

2.2. Model: AWD-LSTM

For multi-class classification, we have used Average Stochastic Gradient Descent Weighted Dropped Long Short Term Memory (AWD-LSTM) model. The brief structural description of this model is given in the *supplementary S2*. We have trained and validated this AWD-LSTM with the input and output vectors having the same length. For multi-class classification, we have replaced the 'decoder' layer of AWD-LSTM by some fully-connected layers, known as custom head. The encoder in first block gets trained during the first phase and acts like a pre-trained encoder for the second block. It is better than random initialization, and the networks already contains a lot of semantic information about the input data. ***The high-level idea of the present protocol is to combine a pre-trained encoder with the 'custom head' to obtain a better classification: this is motivated by ULMFIT as implemented for the NLP application as articulated in***[16]. The weights of the Block B (figure 1), copied from

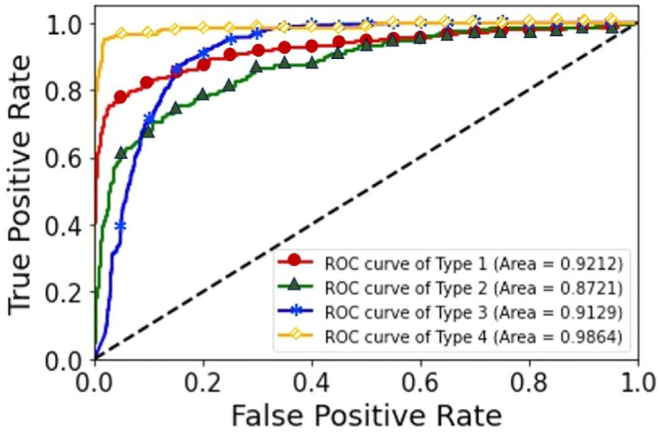

Figure 4. Receiver operating characteristics (ROC) curve.

Table 1. Values of different metrics obtained using AWD-LSTM model and random guess.

Classification Performance Measure	AWD-LSTM (%) Class-wise	Random Guess (%) Class-wise	AWD-LSTM (%) Weighted Average
Accuracy	91.3(Overall)	65.2 (Overall)	—
Recall Score	74.5(Type-1)	14.1(Type-1)	91.0
	19.1(Type-2)	3.6 (Type-2)	
	74.6(Type-3)	2.8 (Type-3)	
	97.9(Type-4)	79.2 (Type-4)	
Precision Score	82.4(Type-1)	14.1 (Type-1)	90.0
	65.6(Type-2)	3.6 (Type-2)	
	94.4(Type-3)	2.8 (Type-3)	
	92.6(Type-4)	79.2 (Type-4)	
F1 score	78.3(Type-1)	14.1 (Type-1)	90.0
	29.6(Type-2)	3.6 (Type-2)	
	83.3(Type-3)	2.8 (Type-3)	
	95.2(Type-4)	79.2 (Type-4)	

We achieved a higher weighted average *F1* score than obtained in [17].

the pre-trained encoder in Block A are frozen for initial 4 epochs. Thereafter, all the weights are unfrozen and the whole network is trained. The training details of the model can be found in the *supplementary S3*.

3. Results

We have analyzed the performance of AWD-LSTM method on the dataset of 40,877 opcodes by calculating the accuracy, precision, recall, F1 score, and confusion matrix (discussed in *supplementary S4)*. The variation of *F1* score during training (figure 2) indicates that the model quickly achieve very high *F1* score. The model is trained for 132 epochs. The curve has fluctuations but overall it increases for higher epochs. The values of these parametersdifferent metrics used for performance measurement of our mod areel is tabulated in Table 1.

The diagonal elements of the confusion matrix, C ($^{C}1,1$, $^{C}2,2$, $^{C}3,3$, $^{C}4,4$) obtained using AWD-LSTM method as well as using random guess (figure 3), are the numbers of correctly classified SCs. The rest are the misclassified SCs. The confusion matrix depicts that the model correctly classified a minimum of 74% for all types except for Type-2. The prodigal vulnerability evades appropriate detection. Whether this is because of some technical nature of the Type-2 vulnerabilities or due to some weakness in the present scheme is unclear and deserves further investigation.

Receiver Operating Characteristics (ROC) curve is the plot between true positive rate and false positive rate of predictions from a neural network for various classes (figure 4). If the data point belongs to Type 1 and the prediction is other than Type 1, it is considered as a false positive. The Area Under Curve (AUC) metric indicates the neural networks' capability to distinguish between various classes. The best performance is observed for Type-3 (AUC $>$ 99%), while the minimum, observed for Type-2 (AUC $>$ 98%), is still very good.

4. Conclusion

The adaptation of pre-trained neural networks is increasing in diverse areas of deep-learning applications as they are proven to be useful in achieving better performance. The idea is to combine a pre-trained encoder with the 'custom head' (section 2.2) to obtain a better classification motivated by ULMFIT. In the present article, we have presented similar adaptation for multi-class classification for the SCs where we have used two neural networks where the first network learns a significant amount of semantic information about the input data helping the second network to achieve better performance. This method produces acceptable results with an accuracy of 91.0% and an *F1* score of 90.0% in multi-class classification of SCs. The high AUC indicates robust performance of the algorithm for the detection of vulnerabilities in the SCs. The performance of AWD-LSTM model is compared with the random guess method and it is shown that the model is far superior to the random guess method. This model can be utilized for development of smart contract security threat detection tools which can be scalable with the number of SCs. Though the detection performance of the current model is acceptable, it solely depends on how correctly the SCs used for training the model are classified. Thus in future, the challenge of producing more number of accurately classified SCs need to be addressed in order to further improve the efficiency of the model. We also outlined the scope and direction for future research for improved performance.

Acknowledgments

We thank Sourav Sen Gupta, Wesley Joon-Wie Tann, Xing Jie Han and, Yew-Soon Ong for sharing the preprocessed SC data with us, which was used for this study. The authors thank Ms. Shubhangi Sinha for careful reviewing of the manuscript.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.https://github.com/AjayKumarGogineni777/Smart-Contract-analysis-using-AWD-LSTM.

Data availability

For the data used in the paper, one will need to contact our collaborators Prof. Sourav Sengupta at sg.sourav@ntu.edu.sg or the corresponding author of the present article, Dr Kisor Kumar Sahu at kisorsahu@iitbbs.ac.in.

ORCID iDs

Kisor K Sahu https://orcid.org/0000-0003-2488-0420
Raj Kishore https://orcid.org/0000-0003-0843-8763

References

[1] Szabo N 1996 Smart contracts: building blocks for digital markets *EXTROPY: The Journal of Transhumanist Thought* **18** 2
[2] Antonopoulos A M and Wood G 2018 *Mastering ethereum: building smart contracts and dapps* 1st (USA: O'reilly Media Inc) https://www.oreilly.com/library/view/mastering-ethereum/9781491971932/9781491971949
[3] The Ultimate Guide to Rootstock Blockchain https://blockgeeks.com/guides/rootstock-blockchain/
[4] Buterin V 2013 Ethereum: a next generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper
[5] Hacking Distributed 2016 Analysis of the DAO exploit http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/ [Online; accessed 20-November-2019]
[6] Parity Technologies 2017 Security Alert. https://paritytech.io/security-alert-2/ [Online; accessed 20-Novemebr-2019]
[7] Erickson B J, Korfiatis P, Akkus Z and Kline T L 2017 Machine learning for medical imaging *Radiographics* **37** 505–15
[8] Huang X, Shan J and Vaidya V 2017 Lung nodule detection in CT using 3D convolutional neural networks *2017 IEEE 14th Int. Symp. on Biomedical Imaging (ISBI 2017)* pp 379–83(IEEE)
[9] Ishii T, Nakamura R, Nakada H, Mochizuki Y and Ishikawa H 2015 Surface object recognition with CNN and SVM in Landsat 8 images *2015 14th IAPR Int. Conf. on Machine Vision Applications (MVA)* pp 341–4*(Piscataway, NJ)* (IEEE)
[10] Liu Y, Zhao T, Ju W and Shi S 2017 Materials discovery and design using machine learning *Journal of Materiomics* **3** 159–77
[11] Liu R, Kumar A, Chen Z, Agrawal A, Sundararaghavan V and Choudhary A 2015 A predictive machine learning approach for microstructure optimization and materials design *Sci. Rep.* **5** 11551
[12] Li X *et al* 2016 Empirical analysis: stock market prediction via extreme learning machine *Neural Computing and Applications* **27** 67–78
[13] Du M, Li F, Zheng G and Srikumar V 2017 Deeplog: anomaly detection and diagnosis from system logs through deep learning *Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security* pp 1285–98*(New York)* (ACM)

[14] Shen Y, Mariconti E, Vervier P A and Stringhini G 2018 Tiresias: predicting security events through deep learning *Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security* pp 592–605*(New York)* (ACM)
[15] Shin E C R, Song D and Moazzezi R 2015 Recognizing functions in binaries with neural networks *24th {USENIX} Security Symp. ({USENIX} Security 15)* pp 611–26
[16] Howard J and Ruder S 2018 Universal language model fine-tuning for text classification arXiv preprint arXiv 1801.06146
[17] Tann A, Han X J, Gupta S S and Ong Y S 2018 *Towards safer smart contracts: a sequence learning approach to detecting vulnerabilities* https://arxiv.org/abs/1811.06632
[18] Nikolić I, Kolluri A, Sergey I, Saxena P and Hobor A 2018 Finding the greedy, prodigal, and suicidal contracts at scale *Proc. of the 34th Annual Computer Security Applications Conf.* pp 653–63*(New York)* (ACM)
[19] Wood G 2014 Ethereum: a secure decentralised generalised transaction ledger *Ethereum Project Yellow Paper* **151** (2014) 1–32