Machine Learning for Smart Contracts

Eren Akgunduz

CAP 5625

April 23, 2023

1 Introduction

Vending machines provide people around the world with snacks and refreshments on a daily basis, and for the vast majority of people the process of selecting an item, paying for it, and receiving said item at one of these is a trivial one. Assuming the machine functions correctly, not many find it a worthwhile exercise to even give the endeavor a second thought, and it is perfectly understandable why. After all, the machinery is designed to ideally complete its main tasks of receiving the user's payment and providing them with their item of choice without dilemma that would serve to draw additional attention upon itself. For a broken vending machine, additional parties may get involved to repair it, or replace it depending on its status, but beyond that circumstance not much in the way of human maintenance or interference occurs at all. Yet, whether the beneficiaries of these sorts of systems actively give thought to how exactly the machine they decided to use for buying an ice cream sandwich actually operates with respect to its programming or not, transactions of this sort and the manners in which they are automated act as a fundamental analogy for an entirely different paradigm, but one which shares many of the same key concepts at its core: smart contracts. It is worth pointing out that the man known for popularizing and advocating for the concept of a smart contract, and for introducing it to the general public in a way that predicated their widespread adoption for conducting and managing cryptocurrency transactions in a fair and effective manner, Nick Szabo, himself utilized the vending machine metaphor while in the midst of developing the ideas central to what a smart contract entails (ethereum.org n.d.). Often, images of legal paperwork and formal agreements between what could be various explicitly human parties — in many cases involving representatives and intermediaries — come to mind, when in reality we can adequately and sufficiently describe a contract as simply an agreement of some sort. With that established, and with one of the implicit intentions of a smart contract framework being to either alleviate or completely eliminate the flaws associated with the aforementioned strategies, it becomes possible to assign a definition to the smart contract. Far from a nebulous idea, smart contracts are absolutely prevalent in a practical sense in today's world, arguably to an extent as ubiquitous as that of vending machines. Essentially, they are nothing more than actual digitized agreements in the form of computer code which executes in a particular way, based on the expressed terms of the agreement, once the conditions of the agreement are realized (Ethereum n.d.).

2 Background

Particularly valuable in the context of blockchain-based transactions conducted using some applicable cryptocurrency, smart contracts allow for all the relevant terms, conditions, and truly the entirety of the logic factoring into a blockchain-driven transaction to reside in a script written in a programming language specifically designed for the purpose of facilitating these — in essence completely eliminating the need for a centralized authority or intermediaries to validate and approve of these procedures manually. In addition, they maximize transparency when it comes to the deal instated by the contract — making any ambiguity or possibility for varying interpretations that could lead to issues, and also the element of trust which plays a critical role in the upholding of human contracts and transactions, obsolete (Ethereum n.d.). Furthermore, the depersonalization of contracts by way of being in "smart" form plays a massive role in the reduction of potential risks and costs that can be seen with real people in the picture (Zheng et al. 2020). Chief examples of languages which make the automation inherent to smart contracts possible include Solidity and Vyper for the Ethereum blockchain. Dependence on a virtual machine for compilation is far from an uncommon situation in the case of numerous programming languages, and the aforementioned are no exception, as they take advantage of the Ethereum Virtual Machine (EVM) in order to run and get deployed (ethereum.org n.d.). Despite the existence of nefarious applications for smart contracts such as Ponzi schemes, various examples of legitimate and beneficial use cases for smart contracts exist, which make their proper implementation and resilience against any potential attempts to attack or compromise parties using the very code which so many rely on for a number of respectable purposes that much more important (Zheng et al. 2020). As such, it is no wonder that even well before the writing of valid and functioning smart contract code recently became feasible on a large scale with the rollout of large language model (LLM)-driven chatbots employing deep learning transformers and natural language processing — providing even individuals with little to no experience with programming, or at least with the particular languages in question, with the capabilities and opportunity to generate effective smart contracts quickly, as long as due diligence is still done — the primary focus of the literature lying at the confluence of smart contracts and machine learning concerned using ML algorithms and models to help detect and clearly identify vulnerabilities in smart contract code, so that individuals can resolve such issues proactively and deny any chance for exploitation.

3 Concerns: why machine learning?

While extremely valuable, this sort of study has been emphasized to such an extent that other pressing challenges in this area may have been overlooked a bit overall by comparison. Namely, some authors have expressed concern regarding obstacles to human readability in the creation of smart contracts — in particular when their code gets compiled by their respective virtual machines to bytecode — as well as some functional and technical issues. These include potential *re-entrancy* in functions which hackers could use to steal currency in the digital format involved, *block randomness* issues where the ideally random outcome with pseudo-random number generators could be rigged and mindfully manipulated to favor the probability of certain outcomes in applications such as betting pool or lottery-backing contracts, and overcharging of the relevant cryptocurrency due to some suboptimal code implementation (Zheng et al. 2020). Systematic analyses have proposed a set of preventative measures that seek to address exactly these problems (Kushwaha et al. 2022). Besides attacks, scams, or unfair advantages that crypto miners could achieve running contracts concurrently or very quickly, though, a source of need for trust returns when it comes to tackling the issue of supplying data inputs from the outside world to the smart contracts which will process them, since they are unable to do this themselves by design. For this task, many blockchains rely on what are known as *blockchain oracles*. These people, where it's often a pool of people who together vote on a relevant outcome being a certain way and not another, have to be trustworthy, and researchers have discussed certain advances that have helped in achieving this, such as third-parties thought to be reliable as well as decentralization (Zheng et al. 2020).

With all these considerations noted for their respective aspects in which they are critical to the success of smart contracts, we must still recognize that one of their biggest concerns, which researchers around the world are not only aware of, but also actively putting their cognizance of the issue, knowledge, and skills to great use to remedy, has to do purely with how correct, reliable, and vulnerability-proof their code actually is. Verifying how correct a smart contract genuinely is can be a difficult task, but two major strategies have been harnessed to this end: analyzing the bytecode, and analyzing the source code itself if it is available. Machine learning algorithms are a strategy that immediately occurs to anyone looking to do these tasks, and they absolutely lie at the heart of analyzing smart contract code as a result (Zheng et al. 2020). Various approaches have emerged, each with their own set of merits and possible downsides. Therefore, it is worth discussing what exactly these methods are and what they entail in some more detail.

4 Deep learning

Deep learning is a subset and field in machine learning that has witnessed an explosion in activity, interest, and progress over the past decade. One could consider effectively selecting a model utilizing deep neural networks, which are especially susceptible to excessively high variance, and then proceeding to tune it with a set of hyperparameters which seem to fit the task well, to be a highly sought after task due to its benefits, notwithstanding the complexities involved in its successful execution. For instance, a security analysis involving the use of four distinct machine learning classifiers to label 1,013 Solidity smart contracts as either having vulnerabilities present or not after labeling with 46 generated labels using two static code analysis tools together, utilized a five-layer neural network as one of the four. The researchers took into account that their data was imbalanced, so instead of measuring the performance of their models on the test data, with a split of 20%, with the accuracy alone, they also utilized F1 scores. Discovering that the various classifiers, such as the neural network, were best suited for labeling in the case of only particular tasks as opposed to the other labels, but that their overall combined performances ranged from decent to excellent and the computational time and resource intensiveness was much improved compared to existing static code analysis engines, they arrived at the conclusion that a model such as theirs which combined the capabilities of deep learning with several other machine learning classification algorithms was more practical than using static code analyzing engines for large amounts of smart contracts, as opposed to a case-by-case basis (Momeni, Wang, and Samavi 2019). Convolutional neural networks (CNNs), which are a popular deep learning architecture for not only image data, but across various disciplines and data types owing to its unique characteristics and practices of applying filters and convolution to the input to learn to recognize its important traits, was chosen in another paper also seeking to analyze Solidity smart contracts for vulnerabilities. However, they used a much larger dataset of over 8000 smart contracts with a different split (train 60%, validation 20%, test 20%) and labeled with three rather than just two static analysis tools. In addition, they used softmax in their final layer with the aim of further minimizing their categorical cross-entropy loss. However, they had much fewer labels with just three main categories, and observed good accuracy across the board, leaving them satisfied with their results and proposing it as a real solution despite admitting that future attempts could benefit from a more multinomial classification, as well as a more thorough scrutiny under which others in the field would replicate and verify the results of their model under different data and circumstances (Sun and Gu 2021).

Proposals for an authorization system for industrial internet of things (IIOT) device smart contracts that is fully decentralized and based on the autoencoder neural network also exist. The autoencoder deep learning architecture is essentially composed of two connected neural networks, with one serving as the encoder and the other as the decoder. Seeking to reduce the dimensionality and compress the data into an efficiently storable and learnable format before aiming to recreate the original data from the encoded version with the decoder, this strategy makes it an effective tool in the arsenal of anyone looking to classify data effectively in an unsupervised manner, or even to help generate new media. Furthermore, the proposal in question applied the L1 penalty, also known as lasso regression, whilst training the autoencoder, with the penalty term to the cost function as follows:

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

Upon deconstructing the smart contracts into a suitable set of deep learning features, their training yielded accuracies on whether the test data was anomalous or otherwise were very high even when compared with other machine learning models also suited for the task such as an SVM. They concede that while their model performs well, it is subject to the known set of pitfalls of autoencoders, which include computational cost and poor interpretability by humans, while claiming it is an approach that is more similar than other comparable models to the function of actual brain cells in biology (Demertzis et al. 2020). Meanwhile, even hybrid deep learning models to analyze Solidity smart contract bytecode were also tested, yielding F1 scores as high as, and in some cases even higher than, models the researchers compared their own with, mentioning that they wished to improve their feature extraction process and diversify the amount of vulnerability labels their model could handle (Shakya et al. 2022).

4.1 LSTMs

Within the realm of deep learning, recurrent neural networks (RNNs) are also of great interest — where the neuron connections cycle back around and are not just feed-forward. In this way, they present numerous advantages for dynamic behavior and modeling for predictions and beyond. Notorious for its great degree of success with even endeavors considered quite complex by humans is the long short-term memory (LSTM) architecture. This particular kind of recurrent neural network was explored for its applicability to detecting vulnerabilities in smart contracts as well, on several occasions. One such case utilized the average stochastic gradient descent (AWD-LSTM) variation and a fully connected classification network, to categorize its Solidity contracts into one of the four following categories: Normal, Suicidal, Prodigal, and Greedy. These were derived in a matter either characteristic of, or also actively utilizing, natural language processing (NLP). Along with the researchers' usage of ULMFIT, they were able to achieve high F1 performances on their test data, though they acknowledged that the precise and clean labeling they were able to achieve may not realistically be in a position others can always replicate in future cases (Gogineni et al. 2020). Similar to what some other literature, including what we have discussed already, examines in how the paper focuses on internet of things (IOT)-aligned smart contracts, researchers decided to test the fit of a binary LSTM classifier for Solidity bytecode data obtained through a Google BigQuery. Outlining even the necessary psuedocode for their LSTM-based classification model, they achieved high scores with regard to accuracy, precision, and recall, although the results seem somewhat questionable when the other deep learning models they tested (other neural networks and gated recurrent unit models or [GRU]) exhibit extremely similar results. It appears that future attempts could incorporate different validation and testing methods, even though they themselves mention in their conclusion that testing across an even greater variety of tuned models could reveal plenty of insights (Gupta et al. 2022).

5 Other classifiers

Besides deep learning, classification models that were used as viable schemes for a longer span of time as part of the machine learning catalogue are still highly applicable to this day. Therefore, it is no surprise that algorithms such as support vector machines (SVM), Naive Bayes, Decision Tree, and Random Forest were also put to use in the reviewed literature for the task of identifying vulnerabilities and classifying smart contract code as vulnerable or otherwise, even if in some cases only as a frame of reference for comparison with a deep learning model of their choice as was the case with some of the previously discussed works. On the other hand, Naive Bayes was used as the primary classification model in one paper intending to explore a system where blockchain technologies and AI techniques work towards a mutual advancement, such as with improved dataset distribution and reliability among AI researchers, and for using AI to reduce a computational burden on verifying entries to the blockchain ledger. The main task of the researchers involved this manner of "joining forces" by operating a pre-trained Naive Bayes classifier with blockchain components. Prediction testing was to occur on-chain. The idea of decentralized predictions occurring in this way acts as a prototype for a huge stride forward in making smart contracts "smarter" as the title suggests. Although the smart contracts were limited by the lack of floating point calculation on the blockchain, they achieved accuracies that approached native Python implementations of the same Naive Bayes Classifier, showing great promise for potential future applications (Badruddoja et al. 2021).

Returning to the prevalent theme of vulnerability detection, a framework which does not even require viewing the Solidity source code for its analysis of potential vulnerabilities, and instead only the transaction logs, called Dynamit, was utilized by researchers in another paper for assigning labels. They do this with the intention of putting a dynamic analysis pattern to use which takes the actual patterns and practices involving the smart contracts to use, rather than only examining the code itself, where an exploitation of a vulnerability may look as follows:

Example contract that is vulnerable by reentrancy principle (Eshghie, Artho, and Gurov 2021)

```
contract Vulnerable {
  function donate(address to_) public payable
  {
    require(to_.call.value(1 ether)());
  }
}
```

Example contract which exploits the vulnerable one (Eshghie, Artho, and Gurov 2021)

```
contract Attacker {

  Vulnerable public vul_contract;

  function startAttack(address _addr) public

  {

    vulContract = Vulnerable(_addr);

    vulContract.donate(address(this));

  }

  function() public payable

  {

    vulContract.donate(address(this));

  }

}
```

Utilizing fuzzing to prevent overfitting and then following that up with training and testing on various different classification models, such as the parametric logistic regression model, as well as Naive Bayes, K-nearest neighbors, SVM, and random forest. Their labels had a benign and malicious component to it for user contracts in addition to just having a “clean” and “vulnerable” contract component. Random forest classification yielded the best accuracy and F1 performance in this case, as well as the lowest false negative rate (FNR) and second-lowest false positive rate (FPR). For the purposes of their experiment, the researchers made sure to use an extensive amount of randomly generated transaction data for their training and testing so that the results would be less impacted by the high bias associated with insufficient randomization in a way that emulates real-world conditions. Other frameworks and more machine learning models, potentially involving more complex algorithms such as neural networks and deep learning architectures evaluated by other literature, were among the aspects the researchers considered for future extensions of their process (Eshghie, Artho, and Gurov 2021).

6 Discussion and conclusion

Common threads are sometimes difficult to establish even across instantiations of a similar topic area, with pattern recognition at the forefront of both human cognition and artificial intelligence use cases, yet in the case of all the reviewed literature a shared concern with regard to handling and standardizing the data were frequent mentions of the imbalance of the smart contracts used for the data. Some of the strategies for dealing with the imbalanced distribution of the data in the classes included random undersampling (mentioned in two papers), taking the F1 score instead of relying on accuracy alone, considering only some unique permutations of the data in the set, and some others as well. With the common case remaining that most smart contracts as a whole, but also within any given sample for a dataset, are benign or non-vulnerable as opposed to the other way around, it appears that arriving at a more optimal solution to this problem could result in something of a breakthrough for all the research in the community collectively.

Ultimately, the sheer quantity of distinct models with varying complexities that have already been examined to assist in elucidating the best ways to determine which smart contract code could be doomed to fail in terms of vulnerabilities and requires some fixing to address that problem, along with novel ways to augment the development of not only smart contracts, but blockchain technology as a whole, as well as improve the integration and cross-compatibility of machine learning tools and implementations in the context of smart contracts on the blockchain, all appear highly promising. Seeing what the future may hold in terms of addressing concerns mentioned previously, using machine learning for assistance as well, that go beyond just the pure code-wise vulnerability aspect of smart contracts and permeate more so into the highly complex and dynamical system of human society and interactions with how they relate to blockchain transactions and more, such as for all the applications like gaming and secure transmission of currency and data as well as the incorporation of decentralized methodologies for still unexplored avenues in the world around us, can have a profound effect on the work and perspectives of those within the machine learning communities and even further — potentially constructing a bridge between disciplines that have previously gone largely unconsidered.

References

Badruddoja, Syed, Ram Dantu, Yanyan He, Kritagya Upadhayay, and Mark Thompson. 2021. "Making Smart Contracts Smarter." In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1–3. Sydney, Australia: IEEE. https://doi.org/10.1109/ICBC51069.2021.9461148.

Demertzis, Konstantinos, Lazaros Iliadis, Nikos Tziritas, and Panagiotis Kikiras. 2020. "Anomaly Detection via Blockchained Deep Learning Smart Contracts in Industry 4.0." *Neural Computing and Applications* 32 (23): 17361–78. https://doi.org/10.1007/s00521-020-05189-8.

Eshghie, Mojtaba, Cyrille Artho, and Dilian Gurov. 2021. "Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning." In *Evaluation and Assessment in Software Engineering*, 305–12. Trondheim Norway: ACM. https://doi.org/10.1145/3463274.3463348.

Ethereum. n.d. "Smart Contracts." Accessed April 23, 2023. https://ethereum.org/en/smart-contracts/.

ethereum.org. n.d. "Introduction to Smart Contracts." Ethereum Development Documentation. Accessed April 23, 2023. https://ethereum.org/en/developers/docs/smart-contracts/.

Gogineni, Ajay K, S Swayamjyoti, Devadatta Sahoo, Kisor K Sahu, and Raj Kishore. 2020. "Multi-Class Classification of Vulnerabilities in Smart Contracts Using AWD-LSTM, with Pre-Trained Encoder Inspired from Natural Language Processing." *IOP SciNotes* 1 (3): 035002. https://doi.org/10.1088/2633-1357/abcd29.

Gupta, Rajesh, Mohil Maheshkumar Patel, Arpit Shukla, and Sudeep Tanwar. 2022. "Deep Learning-Based Malicious Smart Contract Detection Scheme for Internet of Things Environment." *Computers & Electrical Engineering* 97 (January): 107583. https://doi.org/10.1016/j.compeleceng.2021.107583.

Kushwaha, Satpal Singh, Sandeep Joshi, Dilbag Singh, Manjit Kaur, and Heung-No Lee. 2022. "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract." *IEEE Access* 10: 6605–21. https://doi.org/10.1109/ACCESS.2021.3140091.

Momeni, Pouyan, Yu Wang, and Reza Samavi. 2019. "Machine Learning Model for Smart Contracts Security

Analysis." In *2019 17th International Conference on Privacy, Security and Trust (PST)*, 1–6. Fredericton, NB, Canada: IEEE. https://doi.org/10.1109/PST47121.2019.8949045.

Shakya, Supriya, Arnab Mukherjee, Raju Halder, Abyayananda Maiti, and Amrita Chaturvedi. 2022. "Smart-mixmodel: Machine Learning-Based Vulnerability Detection of Solidity Smart Contracts." In *2022 IEEE International Conference on Blockchain (Blockchain)*, 37–44. Espoo, Finland: IEEE. https://doi.org/10.1109/Blockchain55522.2022.00016.

Sun, Yuhang, and Lize Gu. 2021. "Attention-Based Machine Learning Model for Smart Contract Vulnerability Detection." *Journal of Physics: Conference Series* 1820 (1): 012004. https://doi.org/10.1088/1742-6596/1820/1/012004.

Zheng, Zibin, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. 2020. "An Overview on Smart Contracts: Challenges, Advances and Platforms." *Future Generation Computer Systems* 105 (April): 475–91. https://doi.org/10.1016/j.future.2019.12.019.