Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057368

ARTICLE

Detecting Ethereum Ponzi Scheme Based on Hybrid Sampling for Smart Contract

Yuanjun Qu, Xiameng Si*, Haiyan Kang and Hanlin Zhou

College of Computer Science, Beijing Information Science and Technology University, Beijing, 100192, China

*Corresponding Author: Xiameng Si. Email: sixiameng@bistu.edu.cn

Received: 15 August 2024 Accepted: 18 November 2024 Published: 17 February 2025

ABSTRACT

With the widespread use of blockchain technology for smart contracts and decentralized applications on the Ethereum platform, the blockchain has become a cornerstone of trust in the modern financial system. However, its anonymity has provided new ways for Ponzi schemes to commit fraud, posing significant risks to investors. Current research still has some limitations, for example, Ponzi schemes are difficult to detect in the early stages of smart contract deployment, and data imbalance is not considered. In addition, there is room for improving the detection accuracy. To address the above issues, this paper proposes LT-SPSD (LSTM-Transformer smart Ponzi schemes detection), which is a Ponzi scheme detection method that combines Long Short-Term Memory (LSTM) and Transformer considering the time-series transaction information of smart contracts as well as the global information. Based on the verified smart contract addresses, account features, and code features are extracted to construct a feature dataset, and the SMOTE-Tomek algorithm is used to deal with the imbalanced data classification problem. By comparing our method with the other four typical detection methods in the experiment, the LT-SPSD method shows significant performance improvement in precision, recall, and F1-score. The results of the experiment confirm the efficacy of the model, which has some application value in Ethereum Ponzi scheme smart contract detection.

KEYWORDS

Blockchain; smart contract detection; Ponzi scheme; long short-term memory; hybrid sampling

1 Introduction

With its unique distributed ledger structure, blockchain technology stores the value transfer processes of all cryptocurrency transactions [1]. Blockchain technology has the potential to revolutionize the modern financial system due to its unique characteristics, including decentralization, tamper resistance, and traceability [2]. The introduction of smart contracts and decentralized applications (DApps) on the Ethereum platform has further enhanced blockchain's role as a transformative force in the digital age, often referred to as Blockchain 2.0 [3], attracting numerous investors' attention. Following Bitcoin, Ethereum represents the second-largest digital currency trading platform and the most substantial blockchain in terms of facilitating smart contracts [4]. A smart contract is a computer program stored on the Ethereum blockchain that is designed to be executed in a decentralized manner.

Copyright © 2025 The Authors. Published by Tech Science Press.
This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Once deployed on the blockchain, it can be automatically executed by predefined logic when preset conditions are met, without being controlled by any single entity.

However, due to the complexity of blockchain and imperfect regulation, many unscrupulous elements take advantage of the anonymity of blockchain, the consumer psychology of the investor herd, as well as the auto-execution and high trustworthiness of smart contracts [5], and use it as a breeding ground for fraudulent behavior. The research, from 2013 to 2014, revealed that Bitcoin-based Ponzi schemes amassed a total of over $7 million USD [6]. Additionally, many investors lack the expert knowledge to fully understand the workings and potential risks of smart contracts, making it harder for them to identify fraudulent behavior within them [7]. Worse still, due to the anonymity of blockchain, even if a scheme is exposed, it is difficult to trace and hold the criminals accountable [8].

Due to the source code of smart contracts is often difficult to understand and mostly hidden, existing research has opted for Ponzi scheme detection by compiling the bytecode to obtain the code features, as well as the account features of the investors' money flow during the transaction process. However, the existing methods still have shortcomings in the following aspects. First, a large number of verified Ponzi scheme samples are often needed for detection in smart contracts in Ethereum. However, the actual publicly available Ponzi scheme samples employed to train the detection model in Ethereum's smart contracts are relatively scarce and imbalanced [9]. Most previous studies have used a single expanded sample strategy, such as synthetic minority oversampling technique (SMOTE) or other oversampling techniques, when dealing with imbalanced datasets, and have not considered the problem of potentially undifferentiated introduction of noisy samples.

Ponzi scheme smart contracts are often not identifiable in their initial deployment, but as time passes and transaction activity increases, the pattern of their money flows gradually reveals specific temporal characteristics. Existing research, however, does not fully use this feature. Machine learning models are the most frequently applied detection models for Ponzi schemes. Chen et al. [10] employed a range of classification models, including support vector machines (SVM), decision trees and random forests (RF), to identify instances of Ponzi scheme contracts on the Ethereum platform. The application of these models resulted in a notable enhancement in the efficacy of the classification process. However, traditional machine learning models often do not have the ensemble learning capabilities of deep learning models, and they often face challenges in computational efficiency and storage capacity when processing large-scale datasets. In addition, traditional machine learning models often rely on certain data distributions and features, and making it difficult to capture the implicit relationships and contextual information in the data, which makes traditional machine learning models ineffective in dealing with new types of Ponzi schemes. Therefore, in order to ensure the healthy development of blockchain technology, it is imperative to find a way to detect and prevent Ponzi schemes on Ethereum in a timely and effective manner.

To cope with the above challenges, this paper proposes a detection method called LT-SPSD (LSTM-Transformer smart Ponzi schemes detection) based on hybrid sampling. By integrating the SMOTE-Tomek algorithm and the LT-SPSD model, the constructed feature dataset, based on the account features and code features of the contract shows a significant advantage in detecting Ponzi scheme smart contract fraud, which effectively improves the detection accuracy. The main contributions of this paper are as follows:

- Introducing the SMOTE-Tomek algorithm for hybrid sampling of feature datasets to address class imbalance detection in Ethereum Ponzi schemes.
- We propose LT-SPSD, an LSTM-based Ponzi scheme detection model. LT-SPSD can learn from features of smart contract transaction behaviors, and effectively comprehend the context of the

whole sequence, thereby capturing abnormal patterns in the transaction process and promptly discovering hidden Ponzi scheme contracts.
- A feature fusion strategy with dynamic weighting is used. The contribution of LSTM and Transformer outputs is adjusted dynamically through learnable weighting parameters.

The remaining sections of this work are organized as follows. Section 2 summarizes related work. Section 3 describes the LT-SPSD approach in detail. Section 4 introduces the model architecture. Section 5 discusses the experimental results. Section 6 concludes by outlining future research directions.

2 Related Work

2.1 Research on Ponzi Schemes on Blockchains

A Ponzi scheme can be defined as a type of financial fraud. Its defining characteristic is the utilization of the funds of subsequent investors to provide returns to those who have invested at an earlier stage. This creates the impression of a profitable investment opportunity, which in turn attracts further investors. Typically, the return for investors comes from a portion of the investment amount of their direct recruits downlines, and these downlines need to recruit even more downlines to receive returns. Because this model requires an increasing number of new investors to pay the returns of the upper-level investors, it ultimately encounters a growth bottleneck. When the inflow of new investors is not enough to pay returns, the pyramid collapses. In this process, most investors, particularly those who join the Ponzi scheme later, usually suffer heavy losses.

The harm caused by Ponzi schemes has dramatically extended after the blockchain era began, due to the anonymity of blockchain. In addition, the confluence of smart contracts and Ponzi schemes has led to novel forms of fraud that have a considerable impact on Ethereum. At present, research related to Ponzi scheme contract detection can be divided into three categories: The first category is based on source code inspection, and the earliest method was through human identification. Bartoletti et al. [11] identified Ponzi contracts by manually reviewing open source codes, and analyzing descriptive information, source codes, and transaction records in three dimensions to reveal the common features of these smart contracts. Lu et al. [12] converted the source code into a data flow graph and proposed the first SourceP method to detect smart Ponzi schemes using only smart contract source code as a feature. However, the source code of smart contracts is not always easily accessible and most may be hidden.

The second category relies on feature engineering, extracting features from smart contracts and utilizing data mining methods and machine learning models to achieve Ponzi scheme detection in smart contracts. Jung et al. [13] used data mining methods to detect Bitcoin addresses associated with Ponzi schemes. Chen et al. [14] used account and opcode features with an extreme gradient boosting (XGBoost) classifier, but this method did not consider the temporal and global contextual relationships within the data. Zhang et al. [15] applied bytecode similarity combined with the term frequency-inverse document frequency (TFIDF) approach, but account features were not considered and the bytecode-based methods may struggle when facing obfuscated or complex contract structures. Sun et al. [16] constructed behavioral graphs based on interactions between contracts and used the all path tree edit distance (AP-TED) algorithm to calculate behavioral similarity, but the computational cost was high when dealing with a large number of contracts. Fan et al. [17] used the method of ordinal target statistics to process the account and opcode features and constructed an unbiased residual model detection contract based on the Decision Tree. Zhang et al. [18] innovatively extracted bytecode

features and combined them with user transaction features and opcode features to identify Ponzi schemes using an improved light gradient boosting machine (LightGBM). Jin et al. [19] proposed a generic Heterogeneous Feature Augmentation module combined with a machine learning classifier for capturing heterogeneous information related to account behavior patterns.

The third category is based on deep learning. Traditional machine learning models have limitations when dealing with large scale feature datasets. In contrast, deep learning models are ideal for detecting Ponzi scheme contracts due to their powerful text data processing and feature extraction capabilities. Wang et al. [20] applied LSTM with SMOTE, but traditional LSTM models can face challenges when capturing long-term dependencies and global patterns in sequential data. Chen et al. [21] used a convolution-based edge-enhanced graph neural networks (GNN) and an attention mechanism to classify contract transaction graphs, and extracted node and edge features that capture the unique characteristics of Ponzi schemes. Yu et al. [22] extracted 14 transaction features to build a trading network topology and constructed a model to detect Ponzi scheme detection using graph convolutional network (GCN). Cui et al. [23] used a model combining convolutional neural network (CNN) and bidirectional gated recurrent unit (BiGRU) with an attention mechanism to achieve Ponzi scheme detection in smart contracts.

From another aspect, the work by Chen et al. [24], which introduced the semantic-aware detection Ponzi (SADPonzi) method using semantic-aware symbolic execution, focuses on generating semantic information to identify Ponzi behavior but is limited by the need for precise path generation in the contract, which can be computationally expensive. Although these studies have made important progress in the field of Ponzi scheme detection, there are still several limitations. For instance, there are still problems with data noise when dealing with complex and imbalanced datasets, which affects the generalization ability of the model. In addition, the existing models are still deficient in data learning and feature representation, especially in the limited ability to capture the implicit contextual information and complex relationships between data.

In contrast, our proposed LT-SPSD model overcomes some of these challenges by integrating LSTM and Transformer architectures. LSTM excels at processing time series data and capturing long-term dependencies, while the Transformer's self-attention mechanism effectively captures global information and relationships between different parts of the sequence. This combination allows for a more comprehensive analysis of smart contract behaviors, particularly in detecting intricate Ponzi schemes. Additionally, we incorporate the SMOTE-Tomek algorithm to handle imbalanced datasets while reducing noise, thus enhancing the generalization ability of the model and improving detection accuracy.

2.2 Imbalanced Data Classification

Now, the data level and the algorithm level are the two main areas of attention for research on the imbalanced data classification problem. On the data level, the processing methods entail resampling the original dataset using three major strategies: undersampling, oversampling, and hybrid sampling, which mixes undersampling and oversampling. To reduce the influence of data imbalance on the classifier, the data imbalance ratio is changed. On the algorithm level, it is committed to developing machine learning algorithms that can inherently deal with imbalanced data, mainly including cost-sensitive learning, ensemble learning, and others. By selecting the appropriate classification algorithm and adjusting the evaluation metrics, the performance of the model in dealing with the imbalanced classification problem can be improved. In this paper, we plan to use a hybrid sampling method to address the imbalance problem of the original dataset.

Random undersampling [25] is one of the most basic undersampling approaches, and it can significantly reduce the model's training time. However, randomly rejecting samples may eliminate potentially essential information from the majority class, lowering the model's learning effectiveness. Oversampling, unlike undersampling, does not process majority class samples and instead increases the quantity of minority class samples to improve minority class classification performance. The simplest oversampling method usually involves directly copying samples from the minority class to increase its proportion in the dataset. Although the method is relatively simple to implement, it has the potential to result in model overfitting due to the direct replication of the samples, which may not enhance the diversity of the data set. Therefore, related researchers have proposed the synthetic oversampling technique with fewer classes of samples (i.e., SMOTE) [26]. It is evident that SMOTE can effectively minimize the probability of overfitting as a consequence of random replication of data. Conversely, it is essential to acknowledge that when the dataset comprises a considerable number of noisy points or samples with indistinct boundaries that are challenging to categorize, the efficacy of SMOTE may be diminished. SMOTE may not be able to control these samples efficiently, thus overgeneralizing the noise samples and increasing the overlap between different decision boundary classes. To solve the above problems, a series of improved algorithms for SMOTE have emerged, such as Borderline-SMOTE [27], adaptive synthetic sampling (ADASY) [28], density-based SMOTE (DBSMOTE) [29], and so on.

Most of the main streams of current research use oversampling algorithms to increase the number of minority class negative samples in the sample set, thus avoiding discarding sample information. However, it is often difficult to satisfy the need to increase the number of minority class samples and reduce the number of majority class noise samples using only oversampling methods. In order to achieve better data preprocessing results, a hybrid sampling approach can be used, which can combine the advantages of oversampling and undersampling and aims to optimize the dataset by increasing the number of minority class samples and decreasing the number of majority class noise samples. In contrast, the ADASYN algorithm, while addressing the data imbalance problem by adaptively generating synthetic samples, does not directly deal with noisy data, which may affect the performance of the classifier to some extent. Borderline-SMOTE focuses on generating borderline samples to enhance the model's sensitivity to the class boundaries, but it does not deal with noisy samples directly. DBSMOTE is a density-based variant of SMOTE that takes into account the density information of the samples to generate new samples, but it may face similar challenges in dealing with noise and boundary samples because it focuses on sample density and is not specifically designed to remove noise.

Therefore, the SMOTE-Tomek [30] algorithm is used in this paper. It combines the advantages of SMOTE oversampling and Tomek Links undersampling, reduces the noise samples that may be introduced in the oversampling process, and avoids the expansion of the category space caused by synthetic samples, thus improving the model's generalization ability.

3 LT-SPSD Method

3.1 Technical Framework

As shown in Fig. 1, the technological framework for the LT-SPSD approach is presented. In this paper, based on the sample dataset provided by the public dataset of the XBlock website, which provides the contract addresses of the verified contracts and their corresponding categories (whether it is the Ponzi scheme contract address), the bytecode information of the contracts, as well as the transaction information, is collected. On Etherscan.io, the application programming interface (API)

provided by Ethereum captures the transaction data and bytecode information of the smart contract, formats and saves the obtained data in CSV files, and extracts the corresponding account features and opcodes respectively. The opcode features are constructed by counting the frequency of opcode calls.

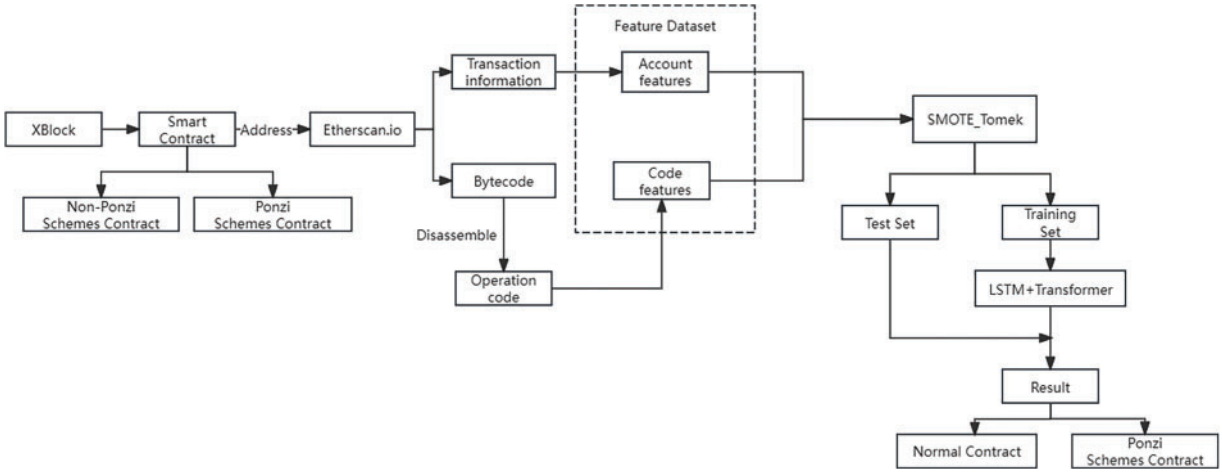

Figure 1: Technical framework of LT-SPSD method

Based on the imbalanced characteristics of the dataset, the dataset will use the SMOTE-Tomke algorithm for sampling preprocessing before use to increase the number of samples in the minority class, thereby improving the generalization ability of the model, while the feature values are compressed to the range of [0, 1] by normalization using the MinMaxScaler function. Then, the dataset is divided into training set and test set according to the ratio of 8:2. The LT-SPSD method uses the LSTM+Transformer algorithm to construct the smart contract detection model for the Ponzi scheme. The training data is fed into the model for training, and the test set is used to validate the model's effectiveness. The LT-SPSD method's effectiveness for detecting Ponzi schemes can be measured using performance metrics such as precision and recall.

3.2 Data Acquisition

In the Ethereum, a transaction is essentially a message passed between accounts, sent from one address to another. Since most interactions on Ether are triggered by transactional behavior, such as the transfer of Ether coins and the creation and execution of smart contracts, this paper chooses to extract features by analyzing transactions. Transactions initiated by user-controlled external accounts are defined as external transactions, while transactions that are initiated by smart contract accounts are defined as internal transactions.

This paper first collects the transaction history of smart contract accounts, as shown in Table 1. "TxHash" is the unique identifier of the transaction, "Block num" refers to the specific block number in which the transaction was confirmed and included in the blockchain. "Timestamp" is the time of the transaction. "From" and "To" respectively represent the sending address and receiving address of the transaction. "Value (ETH)" is the amount of Ether (ETH) sent in the transaction. "ErrCode" means whether the transaction was successfully executed, "Out of gas" indicates that the transaction failed due to insufficient gas, and "None" indicates that the transaction was successfully executed. "Method" refers to the function or method name being called, and "Transfer" is the token transfer function. A

series of key account features can be extracted from these data to identify the transaction patterns and behavioral features of the contract account.

Table 1: Smart contract transaction records

TxHash	Block num	Time stamp	From	To	Value (ETH)	ErrCode	Method
0x2c320b52 ... 642e887f	4168801	1.5E + 09	0x00d3c49e ... 0e84f636	0x0bb3b818 ... 8ebd8f12	10	Out of gas	Transfer
0x123d7c01 ... 5e81ce7c	4168835	1.5E + 09	0x2e8ad99a ... c912bdbb	0x0bb3b818 ... 8ebd8f12	5	None	Transfer

In fact, bytecode exists at the core of Ethereum's smart contract implementation. The process of converting bytecode to opcode is done automatically by the virtual machine and does not directly involve manual conversion by the user. Fig. 2 shows this conversion relationship between the source code, bytecode and opcode (ID: 0x8b7B6C61238088593BF75eEC8FBF58D0a615d30c).

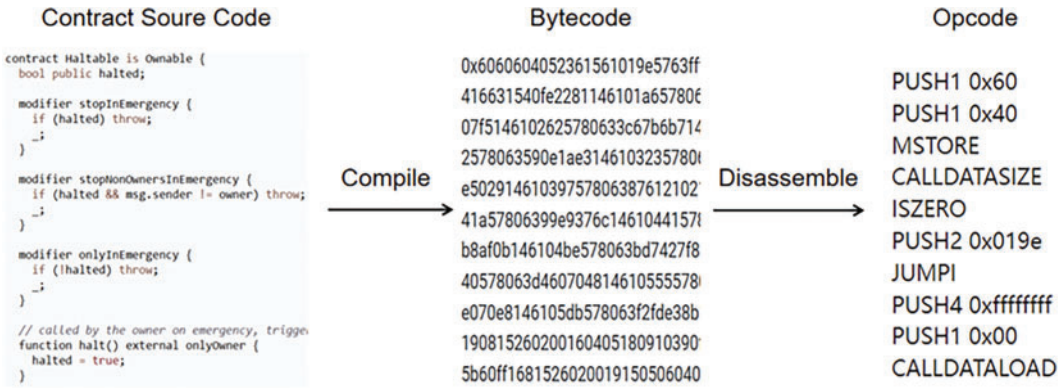

Figure 2: Smart contract code conversion

3.3 Feature Extraction

3.3.1 Account Features

With the addresses in the dataset as a basis, we retrieve relevant transaction data from Ethereum to obtain the account features of the smart contract. Details of all internal and external transactions of the contract can be obtained on Etherscan.io.

Through analysis and observation of the trading behavior of Ponzi scheme contracts, we have identified and extracted 12 key features that exhibit significant variability in contract trading records. Balance are used to reflect the lower balance levels of Ponzi scheme contracts. Total_received, Invested_Before_Receiving, Received_at_least_once, Payment_Rate, Payment_Count, Received_Count, Payment_Received_Ratio are used to reflect the difference in the number of payments and investments made in Ponzi scheme contracts *vs*. ordinary contracts. Max_Payment_Ratio, Most_Payment_Ratio, and Large_to_Small_Ratio are used to reflect the concentration of payment and investment behavior in the contract, as well as the concentration of time

periods. Ponzi contracts usually show a short life cycle, the Contract_Lifetime can be used indirectly to assist in identification.

Table 2 details the statistics of these features, including mean, median and standard deviation.

(1) Balance: the account balance of the smart contract.
(2) Total_Received: the total amount of investment received by the smart contract account.
(3) Invested_Before_Receiving: the proportion of payers who made an investment before obtaining a return.
(4) Received_at_least_once: the proportion of investors who were paid at least once.
(5) Max_Payment_Ratio: the maximum percentage of the contract payment amount.
(6) Most_Payment_Ratio: the largest percentage of contract payments.
(7) Payment_Rate: the rate of contract amount payment.
(8) Payment_Count: the number of transaction expenses for the contract.
(9) Received_Count: the number of investments received by the contract.
(10) Payment_Received_Ratio: the payout rate for the number of contracts.
(11) Large_to_Small_Ratio: the ratio of large to small transactions in a contract (assuming that large transactions are defined as those greater than three times the average transaction amount).
(12) Contract_Lifetime: the life cycle of a contract transaction.

Table 2: Statistical description of account features

	Ponzi			Non-Ponzi		
	Mean	Median	Std	Mean	Median	Std
Balance	1.20	0.00	5.21	126.12	0.00	3579.86
Total_Received	412.33	1.08	2100.65	15,458.28	0.00	305,981.12
Invested_Before_Receiving	0.81	0.87	0.20	0.79	0.88	0.22
Received_at_least_once	0.31	0.14	**0.35**	0.13	0.00	0.25
Max_Payment_Ratio	0.26	0.08	**0.36**	0.20	0.00	0.35
Most_Payment_Ratio	0.26	0.13	0.30	0.19	0.00	0.34
Payment_Rate	0.61	0.78	0.84	12.92	0.00	727.12
Payment_Count	73.28	2.00	323.95	169.07	0.00	1098.14
Received_Count	128.26	10.00	495.75	883.36	6.00	2538.14
Payment_Received_Ratio	0.72	0.20	1.90	0.23	0.00	2.57
Large_to_Small_Ratio	1624.92	0.00	21996.25	1.32E + 12	0.00	5.42E + 13
Contract_Lifetime	0.034470	0.000254	**0.064391**	0.026769	0.000502	0.0551092

A comparison and analysis of the statistical values of the Ponzi scheme contract with those of the normal contract, as presented in Table 2, reveal that there are statistically significant differences between the two contracts for the 12 account characteristics previously mentioned. For example, the mean and median balance values are higher for normal contracts than for Ponzi scheme contracts. This reflects the fact that account balances in Ponzi scheme smart contracts are generally maintained low, while the smaller standard deviation (Std) further confirms that low account balances are more common in such contracts.

Exception for Received_at_least_once, Max_Payment_Ratio and Contract_Lifetime, the other 9 features in the Ponzi scheme contract show very low standard deviation compared to the normal contract, which indicates that there is a high degree of consistency or similarity in the transaction behavior of users in the Ponzi scheme contract. It can thus be concluded that these transaction features serve as reliable indicators for the identification of Ponzi schemes. We employed summary scatter plots for visualization for the three account attributes with non-intuitive distinctions to provide a clearer understanding of the data distribution and trends, as shown in Fig. 3.

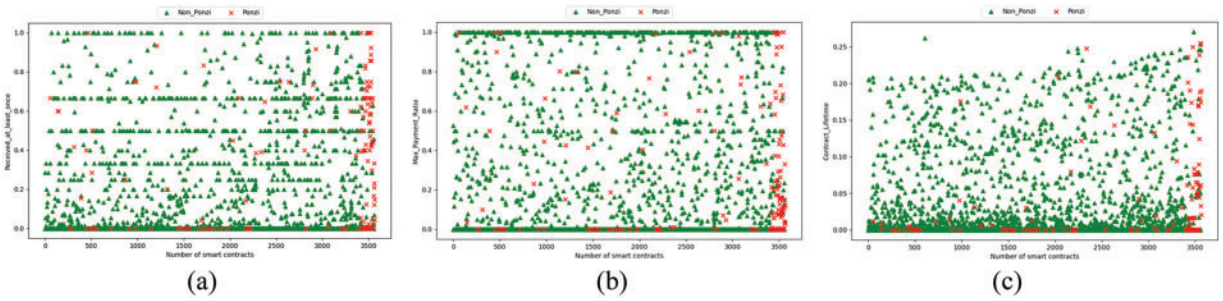

Figure 3: Scatter plot of different account features: (a) Scatter plot of Received_at_least_once; (b) Scatter plot of Max_Payment_Ratio. (c) Scatter plot of Contract_Lifetime

As shown in Fig. 3a–c, the Received_at_least_once of the Ponzi scheme contracts mainly lies between 0.4–0.8, Max_Payment_Ratio mainly lies between 0.1–0.3, and Contract Lifetime is concentrated around 0.06. The Received_at_least_once values, Max_Payment_Ratio values, and Contract_Lifetime values of ordinary contracts are concentrated at 0 or 1 position. As a result, it is possible to use these three account data points as transaction features.

3.3.2 Code Features

For Ponzi scheme smart contracts, although the source code may be carefully designed to conceal the true intent, the opcodes are difficult to completely disguise the underlying mechanism, and its frequency and mode of use are relatively difficult to be completely disguised. Especially when performing operations involving typical Ponzi schemes such as fund transfer and revolving payment, the frequent occurrence of specific opcodes (such as CALL, DELEGATECALL, etc.) becomes a significant identifier, which leads to the differences in the code composition between Ponzi scheme and non-Ponzi scheme contracts.

This paper extracts key features of smart contract code by examining the kinds and frequencies of contract opcodes, exploring the essential distinctions between Ponzi scheme smart contracts and ordinary smart contracts in a timely and effective way. Potentially abnormal behavior patterns can be recognized in part by comparing the frequency and distribution of different sorts of opcodes in Ponzi scheme contracts with regular contracts. After screening, this paper removes some common opcodes that have little impact on distinguishing between Ponzi and non-Ponzi smart contracts and counts the number of occurrences of the remaining opcodes in each smart contract. Based on this statistic, we construct a dataset of 76 code features.

Fig. 4 shows that Ponzi scheme contract opcodes differ significantly from ordinary contract opcodes. For example, Ponzi scheme contracts frequently use fund management type opcodes (e.g., CALL, CALLVALUE, etc.) to implement opaque fund flows, whereas such operations may not be

dominant for ordinary smart contracts. According to the analysis above, it may be able to identify Ponzi scheme smart contracts using opcode features.

Figure 4: Word cloud of smart contract opcode: (a) Opcode word cloud for non-Ponzi scheme contracts; (b) opcode word cloud for Ponzi scheme contracts

3.4 Processing of Imbalanced Feature Data

The feature dataset, which includes 3378 examples of ordinary smart contract characteristics and 186 examples of Ponzi scheme contract characteristics, was created from the code and account features mentioned above. As Fig. 5 shows, the dataset that was acquired has a serious sample data imbalance issue. The ratio of positive and negative samples is about 18:1. To solve the data imbalance problem, this chapter uses the SMOTE-Tomek hybrid sampling algorithm in the preprocessing phase of data preparation. The process of SMOTE-Tomek is as follows:

(1) The imbalanced dataset is divided into two parts: the majority class sample dataset D_{max} and the minority class sample dataset D_{min}.
(2) The SMOTE algorithm is used on the D_{min} to generate synthetic samples.
   a) Randomly select a sample from the minority class D_{min} and determine its k nearest neighbors.
   b) Randomly select N samples from the k nearest neighbors of each minority class sample, and for each of the selected nearest neighbors, perform random linear interpolation with the original minority class sample. The interpolation formula is:

$$\boldsymbol{x}_{new} = \boldsymbol{x}_i + \boldsymbol{rand}\,(\mathbf{0},\mathbf{1}) \times \left(\boldsymbol{x}_j - \boldsymbol{x}_i\right) \tag{1}$$

   where x_i is the original minority sample, and x_j is the selected nearest neighbor and *rand* (0, 1) is a random number in the range [0, 1].
   c) Adding these synthetic samples to the D_{min} in, thus increasing the number of samples in the minority class.
(3) For each majority class sample in the majority class sample dataset D_{max}, finds its nearest minority class sample, and for each minority class sample in the minority class sample dataset D_{min}, finds its nearest majority class sample.
(4) If the distance between the majority class sample and the minority class sample is less than some predetermined threshold, the two samples form a Tomek Links pair.
(5) Delete most of the class samples in the Tomek Links pairs to get a new dataset.

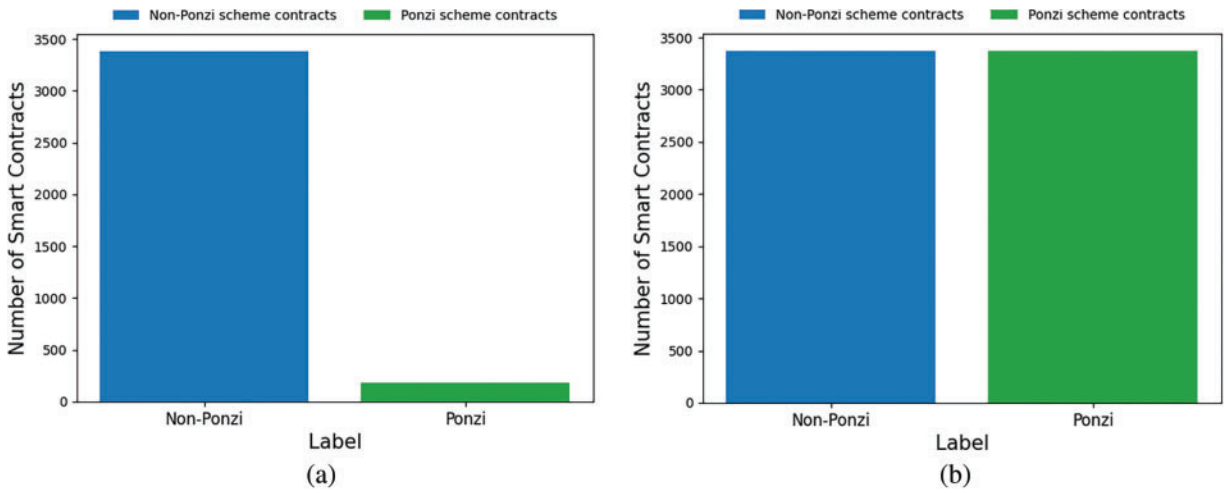

Figure 5: Distribution of data sample categories: (a) Original sample distribution; (b) hybrid sampling sample distribution

4 Model Architecture

In the context of the accelerated evolution of blockchain technology, Ponzi scheme smart contracts bring new challenges to financial security with their hidden and destructive nature, and their inherent features often emerge gradually over time. LSTM has the advantage of dealing with long-term dependencies and data with temporal features, while Transformer captures long-distance dependencies as well as global information through the self-attention mechanism. In this paper, we combine these two techniques to construct a detection model to capture the behavioral features of smart contracts in the time dimension, and then effectively identify Ponzi scheme smart contracts. The architecture of the LT-SPSD is shown in Fig. 6.

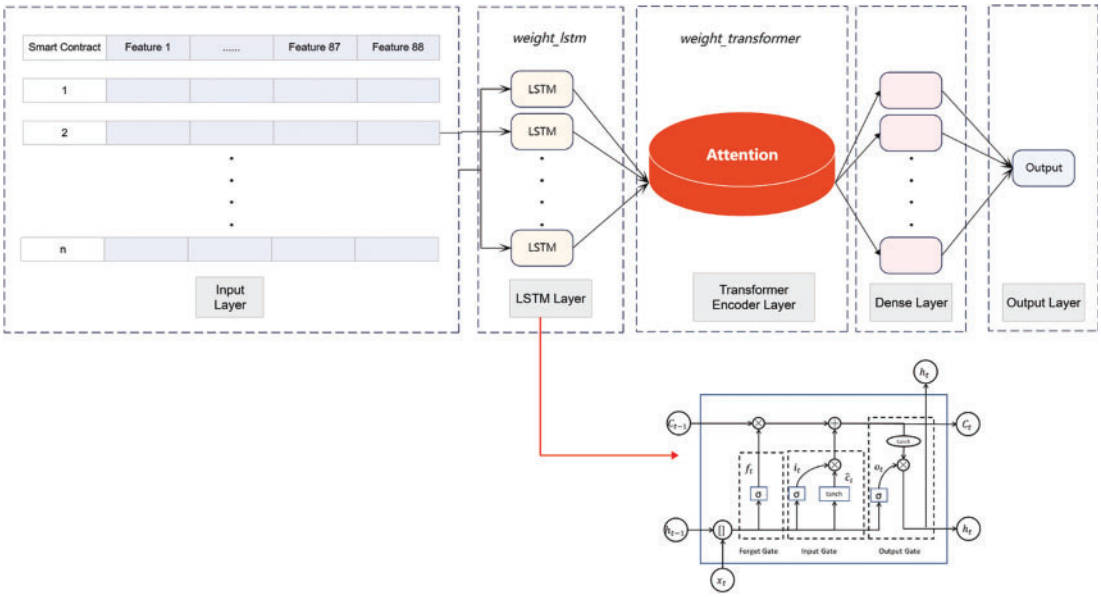

Figure 6: LT-SPSD model architecture

The account features and code features of smart contracts are first processed through the pre-processing phase, including normalization and encoding to ensure the input requirements of these data adaptation models. The data pertaining to the processing of feature information is transmitted to the LSTM layer, which is used for the purpose of acquiring an understanding of the distinctive characteristics exhibited by training concentrated smart contracts. In LSTM, the unit status C_t is used to store and update long-term dependency information in the sequence. The forget gate decides what information to discard from the cell state, the input gate decides what new information needs to be added to the cell state, and the output gate decides the output of the current LSTM cell based on the cell state. Then, the output of the LSTM layer is passed to the Transformer encoder layer, which utilizes a multi-head attention mechanism to process the data. This mechanism maps the input sequence into multiple subspaces, each of which independently performs the self-attention computation to fully capture the multidimensional features of the sequence.

To integrate the advantages of LSTM and Transformer encoder, we adopt a feature fusion strategy, weighted summation. In this process, the output of each layer has a learnable weight parameter, which is *weight_lstm* and *weight_transformer*, which can be dynamically adjusted based on the feedback during the model training process, ensuring that the model can flexibly adjust the contribution of the output of each layer according to the specific needs of the task. Typically, a dense layer is added to the output of the LT-SPSD model to enable additional processing and feature transformation. To determine if a smart contract is a Ponzi scheme contract or not, the predicted value of the model is calculated using a Sigmoid activation function in the classification context. Then, the predicted value is compared with the actual labels of the smart contract to evaluate the model's performance on the training set. This is achieved by calculating a binary cross-entropy loss function.

To mitigate the adverse effects of noise in the data on model training and to enhance the model's generalization capacity, this paper proposes the introduction of a dropout ratio of 0.5 in the LT-SPSD model. This approach serves to diminish the model's susceptibility to the influence of noise and outliers present in the training data. As the model learns on the training set, after each iteration, the model's loss values are provided to the optimizer Adam, which adjusts and optimizes the weights of each hidden layer in the model to increase the model's prediction accuracy for the next round of training. Finally, the smart contracts are used in a test set to determine the detection accuracy of the model.

5 Experiments

5.1 Data Sets

The dataset used in this research is from the open source XBlock dataset. The original dataset comprises 3793 smart contracts, with each contract labeled as either a Ponzi scheme or not. To ensure the accuracy and availability of the dataset, invalid contract addresses were removed by manual inspection in this paper, and contract addresses that cannot obtain transaction data and bytecode data were excluded. The Etherscan.io website was used to crawl the pertinent bytecode and transaction data. Then, by the conversion relationship according to the Ethereum Yellow Paper, the bytecode was transformed into opcodes, with the ratio of each opcode to the total number of opcodes then calculated. In addition, from each contract's transaction data, transaction account features representing smart Ponzi schemes were gained. The final feature dataset comprised account features and code features corresponding to 186 Ponzi scheme contracts and 3378 normal smart contracts. The contract feature dataset included 88 features, of which 76 were opcode context features and 12 were account features. To construct the model, the contract feature dataset was randomly split into a training set (80%) and a testing set (20%).

5.2 Impact of Parameter Epochs

In machine learning, an epoch represents the number of complete training cycles that a model undergoes on an entire training dataset. In other words, one epoch signifies that the model is trained once using the entire training set. However, just using one or a few epochs often doesn't optimize the model, the model may not have fully learned the intrinsic rules of the data. As the number of epochs increases, the frequency of weight updates in the neural network also rises, accompanied by a reduction in the training and test losses of the model. However, when the number of epochs is excessive, the model may be overfitting.

In the LT-SPSD method, we set the epoch to 50. Figs. 7 and 8 show the training and test accuracy of the model displays an upward tendency as the number of epochs increases, accompanied by a reduction in training loss. Once the number of epochs reaches 50, the accuracy and loss values tend to stabilize. The experimental results show that LT-SPSD is highly accurate in its detection capabilities and the phenomenon of overfitting does not happen.

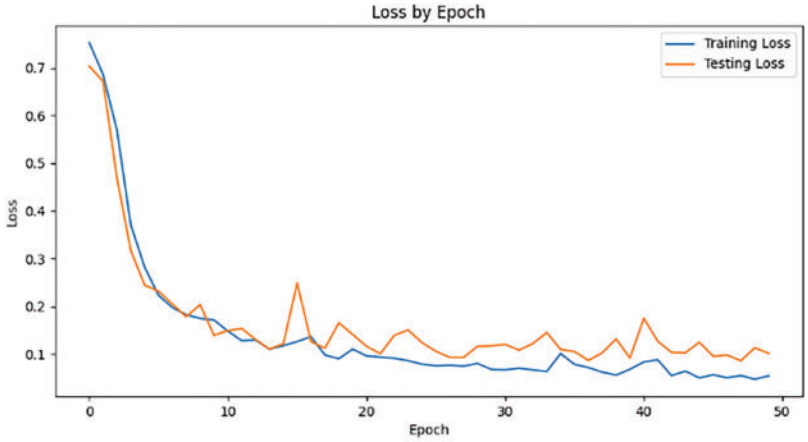

Figure 7: Loss values obtained during 50 epoch

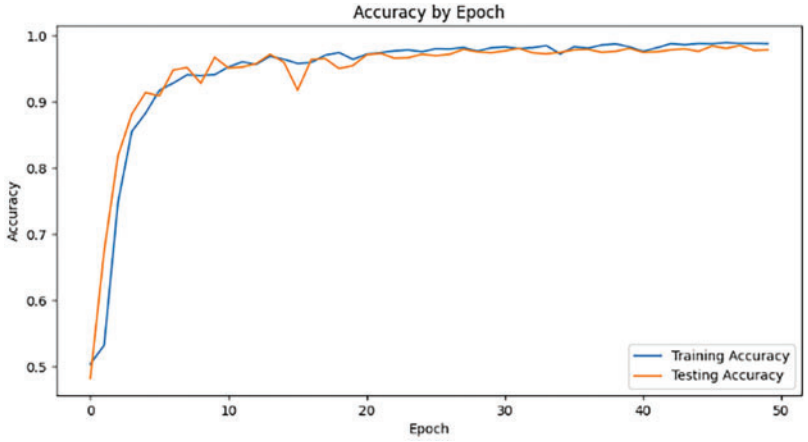

Figure 8: Accuracy values obtained during 50 epoch

5.3 Imapct of Parameter Batch_Size

Batch_size determines the number of samples put into the neural network for processing at one time during model training. Choosing the right batch_size will affect how well the model is optimized and how quickly it converges. A smaller batch_size means fewer samples per iteration, resulting in a longer learning time for the model to converge. As batch_size increases, the training loss decreases, but it also makes the model require more epochs to reach the minimum validation loss, and the model's ability to generalize on the training data decreases. To discuss the effect of the parameter batch_size on the model, we assume that it is chosen in the range (16, 32, 64, 128, 256).

Figs. 9 and 10 show the time consumed during model training and the training accuracy achieved.

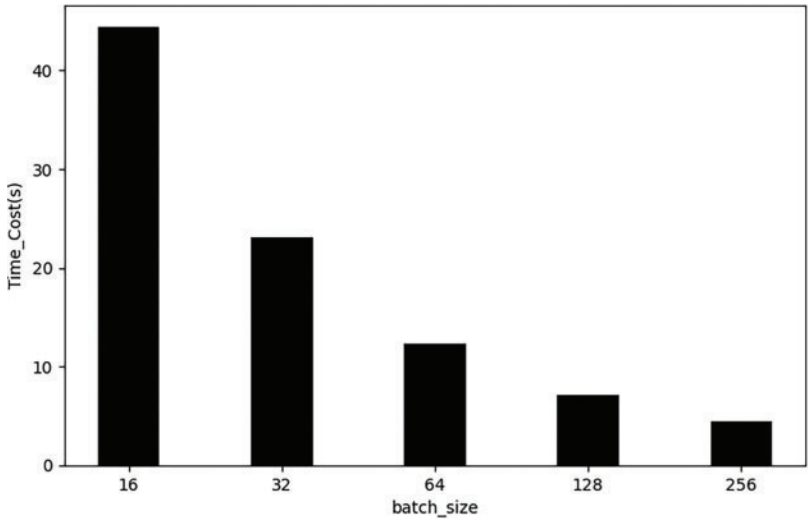

Figure 9: Training time for different number of batch_size

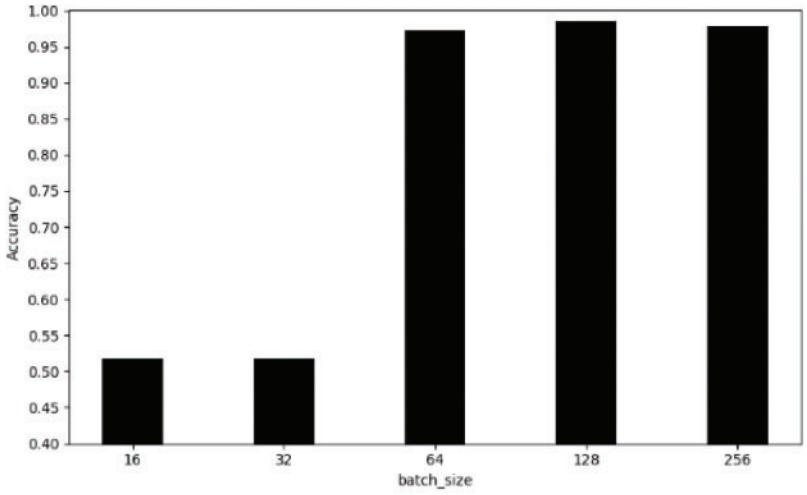

Figure 10: Training accuracy for different number of batch_size

Results show that the time required for training shows a decreasing trend as the batch_size increases. When the number of batch_size is over 128, the change of accuracy is not obvious.

Considering the efficiency of the training time and the accuracy of the model, we finally decided to set the parameter of batch_size to 128 to achieve the best balance between time cost and model performance.

5.4 Impact of Dataset Division

The effectiveness and performance of the model depend on the division ratio of the dataset. A suitable division ratio facilitates more effective training of the model, enables more accurate evaluation of its performance, and allows for understanding of its performance on different data distributions. In this experiment, we adopt a step-by-step incremental method to adjust the size of the test set, and for a fixed sample dataset, we set the ratio of the test set to (0.2, 0.25, 0.3, 0.35, 0.4, 0.45), to investigate the impact of varying test set ratios on model evaluation. Meanwhile, the parameters epoch and batch_size were set to 50 and 128, respectively.

The experimental results show the effect of test set ratio on the accuracy of the model. It can be observed from Fig. 11 that the accuracy of the model varies at different test set ratios. In particular, the accuracy of the model is the highest when the test set ratio is set to 0.2 and 0.35. Keeping a small test set ratio helps to ensure that the training set has sufficient samples to support adequate learning and feature extraction of the model during the training phase. Therefore, we divided the dataset according to the ratio of 8:2.

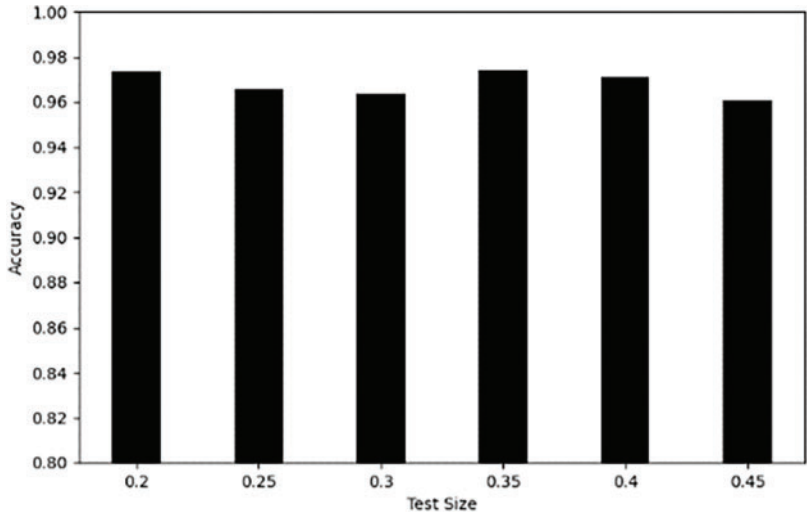

Figure 11: Accuracy for different test set division ratios

5.5 Performance Comparison

To evaluate the effectiveness of our proposed LT-SPSD model, we conducted a comparative analysis with a range of methods. In the experiment, the LT-SPSD model was configured with parameters of epoch = 50, batch_size = 128, and dropout = 0.5. The models of the different methods were trained through the training set. A number of evaluation metrics were introduced to ensure the accurate measurement of the model's efficacy and to facilitate a comparison of its performance with existing methodologies used for the detection of Ponzi schemes with smart contracts. These evaluation metrics included precision, recall and the F1-score. The effectiveness of different detection methods is measured by using these three metrics on the test set.

The definitions of each of these metrics are as follows:

$$\text{Precision} = \frac{truepositive}{truepositive + falsepositive} \tag{2}$$

$$\text{Recall} = \frac{truepositive}{truepositive + falsenegative} \tag{3}$$

$$\text{F1} - \text{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

The results of the experiment are shown in Table 3.

Table 3: Comparative analysis of the performance of different methods

	Precision	Recall	F1-score
XGBoost [31]	0.91	0.67	0.79
RF [32]	0.94	0.64	0.77
RNN [33]	0.94	0.94	0.95
GRU [34]	0.91	0.92	0.92
LT-SPSD	0.97	0.97	0.96

From the experimental results, most neural network methods have superior performance compared to traditional machine learning classification algorithms. For time series data, both XGBoost and RF lack long-term dependency modeling capabilities. Although traditional recurrent neural network (RNN) models are able to capture temporal dependencies, they are prone to the problem of gradient vanishing or gradient explosion, making it difficult to maintain long-term dependencies. However, LSTM-based detection methods, such as LT-SPSD and gated recurrent unit (GRU), show higher accuracy in identifying Ponzi scheme smart contracts. The advantage of these methods is that they can effectively deal with delay effects and long-term dependencies in temporal data. Transformer can effectively capture contextual information in long-range sequential data through the mechanism of multi-head attention, while its parallelized processing capability makes it more efficient in analyzing long sequence data. Among all the compared classification models, our proposed LT-SPSD method shows significant improvement in all performance metrics. In particular, the F1-score of the LT-SPSD method reaches 0.96, which highlights its high efficiency and accuracy in Ponzi scheme smart contracts detection.

This paper uses the SMOTE-Tomek algorithm to achieve data balance. To emphasize the significance of the hybrid sampling technique, we have also presented the confusion matrices with and without the technique in Fig. 12. The SMOTE-Tomek algorithm was used to achieve a balanced distribution within the dataset, which resulted in the model identifying 655 samples as smart Ponzi schemes and only 4 samples as non-Ponzi schemes, with the remaining samples correctly classified. In the event of a highly imbalanced data distribution, the model identified all smart contracts as non-Ponzi scheme smart contracts, leading to an invalid identification. The experimental results show that the using a hybrid sampling technique can effectively address this issue, facilitating the model's enhanced ability to discern the data laws of Ponzi scheme smart contracts and enhance the efficiency of detection.

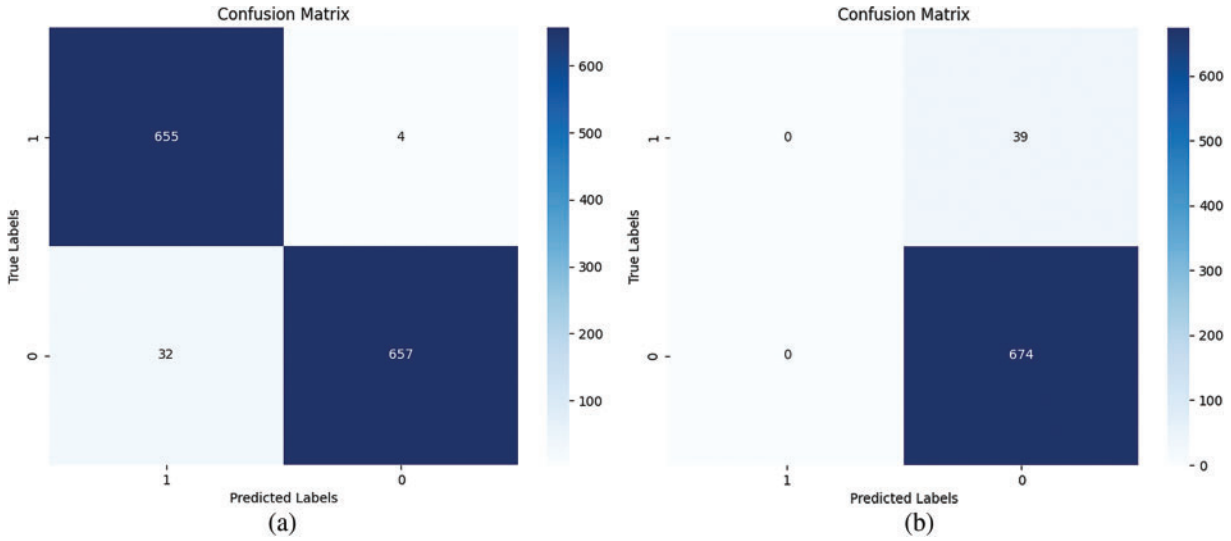

Figure 12: Comparison of confusion matrices: (a) The confusion matrix of model after hybrid sampling; (b) the confusion matrix of model without hybrid sampling

5.6 Computational Complexity

In this section, we conducted a series of experiments to record the execution time of each model on the same dataset. All the experiments are implemented in Python on a PC equipped an NVIDIA A10 GPU, 8.0 GB RAM, running windows 10 × 64 Professional OS. Due to the more complex design of the LT-SPSD method, which integrates more parameter layers, its execution time is longer compared to the other methods. However, it has higher accuracy in detecting Ponzi schemes, which may be at the expense of a certain time efficiency. The average detection time of different methods is shown in Fig. 13.

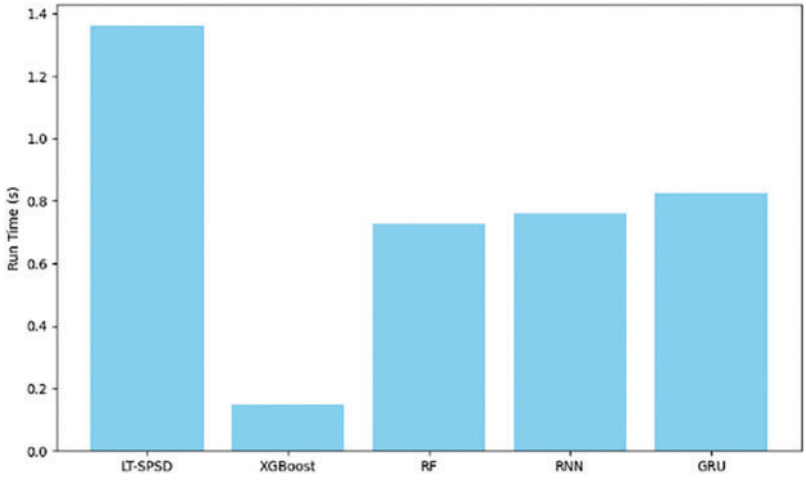

Figure 13: Model execution time

6 Conclusion

Given the Ponzi scheme problem on the blockchain, this paper proposes a model named LT-SPSD that combines account information and code features of smart contracts. The adaptability and performance of the model are improved by fusing LSTM and Transformer encoder by dynamically weighted summation. Furthermore, the SMOTE-Tomek algorithm is used to enhance the learning of minority classes through the integration of oversampling and undersampling techniques. The experimental results showed that the LT-SPSD model performed significantly better in identifying smart contracts associated with Ponzi schemes. This was evident in a number of critical evaluation metrics, such as precision, recall, and F1-score, where there was a noticeable improvement. As a result, it is determined that the LT-SPSD model has some value in detecting Ethereum Ponzi scheme smart contracts.

In future work, we will investigate more advanced techniques, such as the characteristics of generative adversarial networks (GAN) in generating approximations to real data, which can be considered for balancing smart contract feature data. Learning to differentiate between normal contract behavior and Ponzi scheme behavior through the reward and punishment mechanisms of reinforcement learning (RL) and dynamic adjustment of detection strategies to adapt to changing Ponzi scheme strategies. Given the growing number of smart contracts on the Ethereum platform, we will work on researching and developing detection models that can adapt to large-scale datasets.

Acknowledgement: We extend our heartfelt thanks to the participants who graciously dedicated their time and effort to our research. Their contributions were invaluable to the success of our study. Our appreciation also thanks to the funding agency that provided the essential resources necessary for the conduct of this research. Their support was instrumental in bringing this project to fruition. Finally, we are grateful to the anonymous reviewers for their insightful comments and suggestions. Their constructive feedback significantly enhanced the quality of this manuscript.

Funding Statement: This work was granted by Qin Xin Talents Cultivation Program (No. QXTCP C202115), Beijing Information Science and Technology University; the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing Fund (No. GJJ-23), National Social Science Foundation, China (No. 21BTQ079).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Yuanjun Qu and Xiameng Si; data collection: Yuanjun Qu; analysis and interpretation of result: Yuanjun Qu; draft manuscript preparation: Yuanjun Qu, Xiameng Si, Haiyan Kang and Hanlin Zhou. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data openly available in a public repository. The data that support the findings of this study are openly available in XBlock at https://xblock.pro (accessed on 09 March 2024).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, and X. Du, "Scalable and redactable blockchain with update and anonymity," *Inform. Sci.*, vol. 546, no. 15, pp. 25–41, 2021. doi: 10.1016/j.ins.2020.07.016.

[2] P. Zhu, J. Hu, X. Li, and Q. Zhu, "Using blockchain technology to enhance the traceability of original achievements," *IEEE T. Eng. Manage.*, vol. 70, no. 5, pp. 1693–1707, 2023. doi: 10.1109/TEM.2021.3066090.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," in *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[4] Z. Gu, D. Lin, and J. Wu, "On-chain analysis-based detection of abnormal transaction amount on cryptocurrency exchanges," *Physica A: Stat. Mech. Appl.*, vol. 604, no. 7553, 2022, Art. no. 127799. doi: 10.1016/j.physa.2022.127799.

[5] W. Al-Sharu, M. Qabalin, M. Naser, and O. Saraerh, "A secure framework for blockchain transactions protection," *Comput. Syst. Sci. Eng.*, vol. 45, no. 2, pp. 1095–1111, 2022. doi: 10.32604/csse.2023.032862.

[6] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *Financial Cryptography and Data Security*, R. Böhme, T. Okamoto, Eds., Berlin, Heidelberg: Springer, 2015, pp. 44–61. doi: 10.1007/978-3-662-47854-7_4.

[7] T. Moore, J. Han, and R. Clayton, "The postmodern ponzi scheme: Empirical analysis of high-yield investment programs," in *Lecture Notes in Computer Science*, A. D. Keromytis, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7397, pp. 41–56. doi: 10.1007/978-3-642-32946-3_4.

[8] G. B. Mermer, E. Zeydan, and S. S. Arslan, "An overview of blockchain technologies: Principles, opportunities and challenges," in *2018 26th Signal Process. Commun. Appl. Conf. (SIU)*, 2018, pp. 1–4. doi: 10.1109/SIU.2018.8404513.

[9] Y. Chen, H. Dai, X. Yu, W. Hu, Z. Xie and C. Tan, "Improving Ponzi scheme contract detection using multi-channel TextCNN and transformer," *Sensors*, vol. 21, no. 19, 2021, Art. no. 6417. doi: 10.3390/s21196417.

[10] W. Chen, Z. Zheng, E. C. -H. Ngai, P. Zheng, and Y. Zhou, "Exploiting blockchain data to detect smart Ponzi schemes on ethereum," *IEEE Access*, vol. 7, pp. 37575–37586, 2019. doi: 10.1109/ACCESS.2019.2905769.

[11] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting Ponzi schemes on ethereum: Identification, analysis, and impact," *Future Gener. Comput. Syst.*, vol. 102, no. 3–4, pp. 259–277, 2020. doi: 10.1016/j.future.2019.08.014.

[12] P. Lu, L. Cai, and K. Yin, "SourceP: Smart Ponzi schemes detection on ethereum using pre-training model with data flow," 2023, *arXiv:2306.01665*.

[13] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge, "Data mining-based ethereum fraud detection," in *2019 IEEE Int. Conf. Blockchain (Blockchain)*, 2019, pp. 266–273. doi: 10.1109/Blockchain.2019.00042.

[14] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng and Y. Zhou, "Detecting Ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proc. 2018 World Wide Web Conf. World Wide Web-WWW '18*, Lyon, France, ACM Press, 2018, pp. 1409–1418. doi: 10.1145/3178876.3186046.

[15] Y. Zhang, S. Kang, W. Dai, S. Chen, and J. Zhu, "Code will speak: Early detection of Ponzi smart contracts on ethereum," in *2021 IEEE Int. Conf. Serv. Comput. (SCC)*, 2021, pp. 301–308. doi: 10.1109/SCC53864.2021.00043.

[16] W. Sun, G. Xu, Z. Yang, and Z. Chen, "Early detection of smart Ponzi scheme contracts based on behavior forest similarity," in *2020 IEEE 20th Int. Conf. Softw. Qual., Reliab. Secur. (QRS)*, 2020, pp. 297–309. doi: 10.1109/QRS51102.2020.00047.

[17] S. Fan, S. Fu, H. Xu, and X. Cheng, "Al-SPSD: Anti-leakage smart Ponzi schemes detection in blockchain," *Inform. Process. Manag.*, vol. 58, no. 4, 2021, Art. no. 102587. doi: 10.1016/j.ipm.2021.102587.

[18] Y. Zhang, W. Yu, Z. Li, S. Raza, and H. Cao, "Detecting ethereum Ponzi schemes based on improved LightGBM algorithm," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 2, pp. 624–637, 2022. doi: 10.1109/TCSS.2021.3088145.

[19] C. Jin, J. Jin, J. Zhou, J. Wu, and Q. Xuan, "Heterogeneous feature augmentation for Ponzi detection in ethereum," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 69, no. 9, pp. 3919–3923, 2022. doi: 10.1109/TCSII.2022.3177898.

[20] L. Wang, H. Cheng, Z. Zheng, A. Yang, and X. Zhu, "Ponzi scheme detection via oversampling-based long short-term memory for smart contracts," *Knowl. Based Syst.*, vol. 228, no. 6, 2021, Art. no. 107312. doi: 10.1016/j.knosys.2021.107312.

[21] Y. Chen, B. Li, Y. Xiao, and X. Du, "PonziFinder: Attention-based edge-enhanced ponzi contract detection," *IEEE Trans. Rel.*, pp. 1–15, 2024. doi: 10.1109/TR.2024.3370734.

[22] S. Yu, J. Jin, Y. Xie, J. Shen, and Q. Xuan, "Ponzi scheme detection in ethereum transaction network," in *Blockchain and Trustworthy Systems*, H. -N. Dai, X. Liu, D. X. Luo, J. Xiao, X. Chen, Eds., Singapore: Springer, 2021, pp. 175–186. doi: 10.1007/978-981-16-7993-3_14.

[23] B. Cui and G. Wang, "Ponzi scheme detection based on CNN and BiGRU combined with attention mechanism," in *2024 27th Int. Conf. Comput. Support. Coop. Work Des. (CSCWD)*, 2024, pp. 1852–1857. doi: 10.1109/CSCWD61410.2024.10580692.

[24] W. Chen *et al.*, "Sadponzi: Detecting and characterizing Ponzi schemes in ethereum smart contracts," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 2, pp. 1–30, 2021. doi: 10.1145/3460093.

[25] B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin and N. N. Abdullah, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, T. Herawan, M. M. Deris, J. Abawajy, Eds., Singapore: Springer, 2014, pp. 13–22. doi: 10.1007/978-981-4585-18-7_2.

[26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, 2002. doi: 10.1613/jair.953.

[27] S. Smiti and M. Soui, "Bankruptcy prediction using deep learning approach based on borderline SMOTE," *Inform. Syst. Front.*, vol. 22, no. 5, pp. 1067–1083, 2020. doi: 10.1007/s10796-020-10031-6.

[28] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, 2008, pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

[29] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "DBSMOTE: Density-based synthetic minority over-sampling technique," *Appl. Intell.*, vol. 36, no. 3, pp. 664–684, 2012. doi: 10.1007/s10489-011-0287-y.

[30] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20–29, 2004. doi: 10.1145/1007730.1007735.

[31] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[32] S. Hakak, M. Alazab, S. Khan, T. R. Gadekallu, P. K. R. Maddikunta and W. Z. Khan, "An ensemble machine learning approach through effective feature extraction to classify fake news," *Future Gener. Comp. Syst.*, vol. 117, no. 6, pp. 47–58, 2021. doi: 10.1016/j.future.2020.11.022.

[33] Q. Ni and X. Cao, "MBGAN: An improved generative adversarial network with multi-head self-attention and bidirectional RNN for time series imputation," *Eng. Appl. Artif. Intell.*, vol. 115, no. 1, 2022, Art. no. 105232. doi: 10.1016/j.engappai.2022.105232.

[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv.1412.3555*.