Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.040253

ARTICLE

The Detection of Fraudulent Smart Contracts Based on ECA-EfficientNet and Data Enhancement

Xuanchen Zhou[1,2,3], Wenzhong Yang[2,3,*], Liejun Wang[2,3], Fuyuan Wei[2,3], KeZiErBieKe HaiLaTi[2,3] and Yuanyuan Liao[2,3]

[1]College of Software, Xinjiang University, Urumqi, 830000, China

[2]Key Laboratory of Signal Detection and Processing in Xinjiang Uygur Autonomous Region, Urumqi, 830000, China

[3]Key Laboratory of Multilingual Information Technology in Xinjiang Uygur Autonomous Region, Urumqi, 830000, China

*Corresponding Author: Wenzhong Yang. Email: ywz_xy@163.com

Received: 10 March 2023 Accepted: 19 May 2023 Published: 26 December 2023

ABSTRACT

With the increasing popularity of Ethereum, smart contracts have become a prime target for fraudulent activities such as Ponzi, honeypot, gambling, and phishing schemes. While some researchers have studied intelligent fraud detection, most research has focused on identifying Ponzi contracts, with little attention given to detecting and preventing gambling or phishing contracts. There are three main issues with current research. Firstly, there exists a severe data imbalance between fraudulent and non-fraudulent contracts. Secondly, the existing detection methods rely on diverse raw features that may not generalize well in identifying various classes of fraudulent contracts. Lastly, most prior studies have used contract source code as raw features, but many smart contracts only exist in bytecode. To address these issues, we propose a fraud detection method that utilizes Efficient Channel Attention EfficientNet (ECA-EfficientNet) and data enhancement. Our method begins by converting bytecode into Red Green Blue (RGB) three-channel images and then applying channel exchange data enhancement. We then use the enhanced ECA-EfficientNet approach to classify fraudulent smart contract RGB images. Our proposed method achieves high F1-score and Recall on both publicly available Ponzi datasets and self-built multi-classification datasets that include Ponzi, honeypot, gambling, and phishing smart contracts. The results of the experiments demonstrate that our model outperforms current methods and their variants in Ponzi contract detection. Our research addresses a significant problem in smart contract security and offers an effective and efficient solution for detecting fraudulent contracts.

KEYWORDS

Fraud detection; smart contract; ECA-EfficientNet; Ethereum

1 Introduction

Since 2008, blockchain technology has been rapidly advancing, and cryptocurrencies such as Bitcoin and ETH have gained popularity in financial markets. Ethereum, as a representative of Blockchain 2.0, is quickly developing into a fully-fledged platform. As a result, developers are deploying feature-rich Smart Contracts (SCs) on Ethereum to meet various transaction needs.

As the amount of money traded via SCs increases, dishonest entities are intentionally deploying a significant number of fraudulent SCs on Ethereum [1], taking advantage of users' wallet balances. Four types of fraudulent SCs identified as containing fraud are Ponzi SCs, honeypot SCs, phishing SCs, and gambling SCs [2,3]. However, few detection methods for multi-classification fraud SCs have been developed, and most research has focused solely on detecting Ponzi SCs and phishing SCs [4].

The challenges of automatic fraud SCs detection through deep learning usually include:

1) One of the primary challenges in using deep learning to automatically detect fraud in smart contracts is extracting valid semantic expressions directly from their compiled bytecode. Smart contracts are compiled into bytecode by Ethereum Virtual Machine (EVM) and then deployed to Etherscan. However, only a small percentage of the overall smart contracts on Etherscan can be queried for source code [5].

2) Another challenge is finding a suitable data balancing method for smart contracts. Due to the nature of smart contracts, there is an inherent extreme data imbalance between fraudulent and normal contracts. To address the current problems of deep learning-based fraud detection in smart contracts, a detection method based on data enhancement and ECA-EfficientNet is proposed in this paper. ECA-EfficientNet is an improved deep neural network for image classification. This method maps the bytecode directly to RGB three-channel images and then enhances the dataset by channel swapping. The improved EfficientNet is then applied to train on the dataset.

The main contributions of this paper are as follows:

1) To address the issue of extreme data imbalance in fraud Smart Contract (SC) detection, we propose a new image mapping method that extracts features from the bytecode level. Our method maps bytecodes to three-channel RGB images, and we introduce channel swapping based on these images to oversample the dataset. This approach represents a novel contribution to the field of fraud SC detection.

2) We upgraded the EfficientNet model to improve classification accuracy. Specifically, we refined the ECA module [6] using Global Weighted Average Pooling (GWAP) and integrated it into the EfficientNet model. The resulting ECA-EfficientNet model achieved a precision of 99.1% and a recall rate of 99.1% on the public Ponzi SC dataset, representing a significant improvement in recognition accuracy.

3) We integrated the four most common types of fraud SCs (Phishing, Ponzi, Gambling, and Honeypot SCs) into a self-built dataset, which is a novel contribution to the field. We then applied the ECA-EfficientNet model to conduct multi-classification detection on this dataset, achieving an F1-score of 98.2% and a recall of 98.1%. These results demonstrate the ECA-EfficientNet model's superior robustness and high accuracy.

The rest of the paper is structured as follows: Section 2 introduces the basics of fraud SCs on Ethereum and related work concerning them; Section 3 elaborates on the dataset collection and cleaning, as well as the network structure of the ECA-EfficientNet; Section 4 presents the pre-experimental data, pre-processing, evaluation metric selection, experimental results, and performance comparison with other papers or methods; Section 5 summarizes the entire paper, and more solid implications are drawn and included in the concluding remarks.

2 Related Work

2.1 Smart Contracts

Smart Contracts are computer programs designed to execute contract terms automatically when certain conditions are met, eliminating the need for intermediaries to oversee or enforce the agreement. These contracts are self-executing and are written in scripting languages with Turing completeness, allowing them to perform any computation that can be done by a Turing machine [7–9].

Ethereum SCs are a type of smart contract that is deployed on Ethereum blockchain platform and executed through Ethereum Virtual Machine (EVM). These contracts can perform various functions beyond basic transactions, such as mortgages, crowdfunding, loans, decentralized applications, and more.

To create Ethereum SCs, developers use languages like Solidity, which is specifically designed for creating smart contracts on Ethereum blockchain. Once an Ethereum SC is executed, it is broadcast to the network and validated by network nodes. The validated contract is then added to the blockchain as a block, ensuring the contract's transparency, tamper-proof nature, and enforceability by anyone on the network.

Overall, Ethereum SCs empower the development of decentralized applications that can be trusted to execute without intermediaries, improving efficiency, security, and transparency across various industries.

2.2 Fraud Smart Contract

Ethereum blockchain ecosystem has been plagued by fraudulent Smart Contracts (SCs). Incomplete statistics show that more than 10% of Initial Coin Offerings (ICOs) issued on Ethereum are subject to various types of fraud, including phishing, Ponzi schemes, honeypot, and gambling contract fraud, among others. According to a report by ChainAnalysis, victims of such scams suffered a loss of $225 million in the first half of 2017 [10].

Financial security has become a key issue in the blockchain ecosystem, and this article focuses on four types of SCs—Ponzi, honeypot, gambling, and phishing—that target users' wallet balances [11–13].

Ponzi SCs are modeled after traditional Ponzi schemes, where users transfer funds to the contract, and the funds are distributed among a list of users in a pyramid-like structure, with those at the top receiving the most benefits.

Honeypot SCs are contracts that require a certain balance to be transferred before they can be exploited, leading users to transfer funds to take advantage of the "loophole," and thereby cheating users' wallet balances. Phishing SCs are integrated with phishing behaviors, which are deployed on Ethereum and use SCs to deceive users. With the diverse functions of Ethereum SCs, different variants of traditional phishing websites and phishing emails have emerged, which have been integrated with SCs.

Gambling SCs are smart contracts that facilitate traditional offline gambling using SCs on Ethereum. These contracts can have a more significant impact than offline gambling, given the distributed technology of the blockchain.

All types of fraudulent SCs target the virtual currency in users' wallets, and the ability to detect and prevent such scams effectively is critical to the development of Ethereum. Therefore, financial security measures are needed to curb the deployment of fraudulent SCs and protect users' virtual currency.

2.3 Research on Fraud Smart Contracts

Researchers have proposed various methods to detect different types of fraud in Smart Contracts (SCs), but each method has its limitations. For example, Wu et al. [2] proposed a network embedding method that effectively detects phishing nodes using Ethernet transaction network data, but it is ineffective in detecting fraudulent SCs with unknown or tentative transactions. Similarly, Yu et al. [4] proposed a Graph Convolutional Network (GCN) method for detecting Ponzi SCs using transaction information, but it performs poorly in detecting unknown contracts. Agarwal et al. [3] developed a set of evaluation metrics to correlate vulnerabilities and suspicious SCs, but the accuracy of their method may be inconsistent when new types of fraudulent SCs emerge. Torres et al. [14] introduced a classification method for honeypot SCs that is accurate but requires manual analysis. Chen et al. [15] proposed a Text Convolutional Neural Network (TextCNN) based method combined with Transformer for detecting Ponzi SCs, but this method may cause partial information loss due to decompiling bytecode into Solidity source code. In addition, Osegi et al. [16] proposed an Artificial Neural Network trained by the Simulated Annealing technique (SA-ANN) and a Hierarchical Temporal Memory based on the Cortical Learning Algorithms (HTM-CLA) have been proposed to detect credit card fraud (CCF). The HTM-CLA was found to outperform the SA-ANN and the Long Short-Term Memory ANN (LSTM-ANN) by a factor of 2:1. Furthermore, the HTM technique has also been applied to Short-Term Load Forecasting (STLF) and showed promising performance results compared to existing techniques [17]. Therefore, each method has its own advantages and disadvantages, and there is still room for improvement in detecting various types of fraudulent SCs.

Code visualization has been widely used in malware classification tasks, providing an effective end-to-end detection method that can process data samples efficiently and achieve remarkable classification results [18]. Nataraj et al. [19] proposed a method to convert binary code into grayscale images using pixel values for malware classification, and this approach has received extensive attention from researchers since then. To address the issue of imbalanced data, Cui et al. [20] converted malicious code into grayscale images and utilized the bat algorithm. They then used a convolutional neural network to automatically extract features from the malware images. To obtain better classification results, Cui et al. [21] further improved their work by using the non-dominated sorting genetic algorithm II. Naeem et al. [22] developed a method to classify malware by converting binary malware files into grayscale images using two patterns, local and global.

Similarly, there are similar methods in the field of smart contract detection. Lou et al. [23] proposed an improved convolutional neural network as a detection model for Ponzi schemes in smart contracts. The results showed that the improved convolutional neural network can overcome difficulties in training caused by different lengths of smart contracts' bytecodes. Bian et al. [24] proposed an image-based SCs scam detection method using an attention capsule network focused on Ethereum.

3 Dataset and Model

3.1 Model Framework

This paper outlines a general framework for detecting fraud in smart contracts, which comprises four distinct steps, as illustrated in Fig. 1.

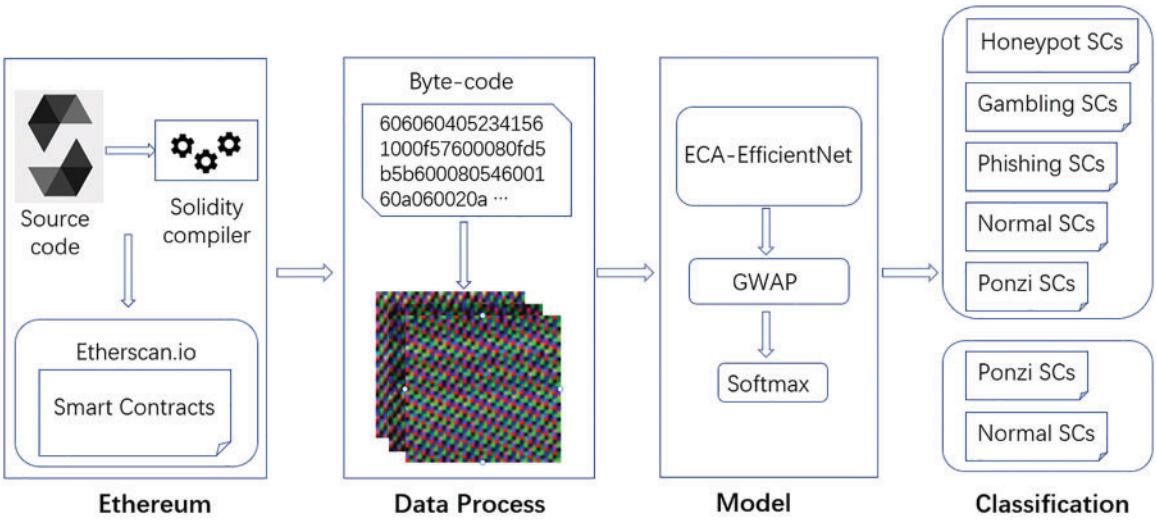

Figure 1: Fraud smart contracts detection framework based on ECA-EfficientNet

1) Initially, we converted the bytecode on Ethereum into color images by assigning grayscale values to each byte in the three RGB channels. We then applied data enhancement to the images by swapping channels to obtain the experimental dataset.

2) Next, we retrieved the RGB images from pre-processing and used them as original features in ECA-EfficientNet. We automatically extracted features that could represent each image category.

3) Subsequently, we employed GWAP instead of a fully connected neural network for the final classification. We compared the computation with the labels to optimize the loss function.

4) Finally, we classified the features using softmax to obtain the final detection results, which included five categories: Ponzi, phishing, gambling, honeypot, and normal smart contracts. Additionally, we performed binary classification of Ponzi and normal smart contracts for Ponzi detection.

3.2 Data Collection and Processing

To begin with, we started by collecting 50 SCs tagged with Ponzi schemes on Etherscan. Further research led us to discover 200 additional Ponzi contract addresses, which were found in the literature [15]. Subsequently, we manually reviewed and cleaned one invalid contract, resulting in the discovery of 249 more Ponzi contract addresses.

Next, we identified SCs labeled as gambling and phishing hacks on Etherscan. However, due to the presence of duplicate and incorrect contracts, we had to manually verify and remove invalid data. Eventually, we obtained the addresses of 133 phishing contracts and 37 gambling contracts.

We also collected the addresses of 285 honeypot contracts through manual analysis by the authors mentioned in the literature [22]. Finally, we gathered the addresses of 3500 normal contracts from the public dataset. Using Python, we were able to crawl the bytecode of the corresponding addresses from Etherscan based on the collected smart contract addresses.

Due to the rarity of fraudulent smart contracts compared to the total number of smart contracts, there is a serious data imbalance problem in the dataset. To alleviate this issue, this study proposes a method of exchanging image channels to increase the number of images and thus augment the amount

of fraudulent smart contract data. To convert the bytecode into images, two steps were necessary, as illustrated in Fig. 2.

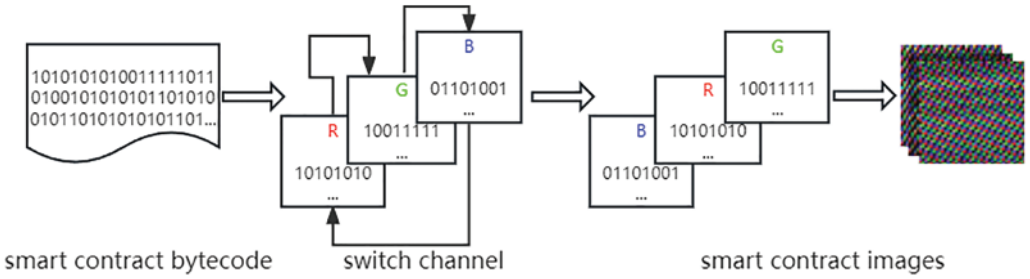

Figure 2: Channel exchange process for RGB images

Firstly, the bytecode was transformed into RGB images by mapping each byte to a corresponding RGB value in three dimensions. Any missing parts were filled with zeroes. Secondly, after obtaining the RGB images, we sequentially swapped the R, G, and B channels of the image to create a new three-channel image.

Channel swapping does not alter the essential features of the image, and the same features as the original image can be extracted in the convolution layer. However, the fully connected layer will show different characteristics from the original image. As demonstrated in Fig. 3, the essential features of the image remain unchanged after channel swapping for the Lenna picture.

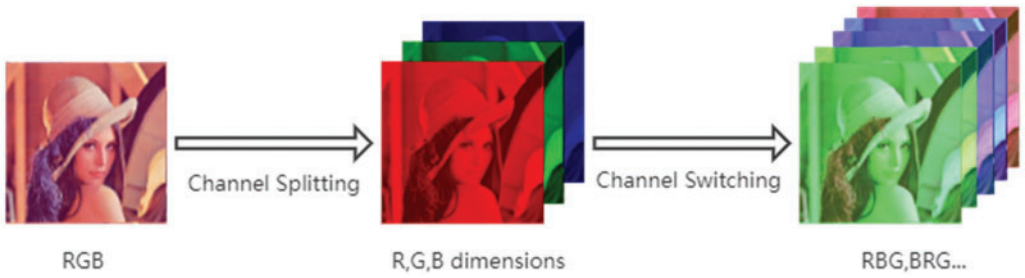

Figure 3: The effect of the Lenna after exchanging channels

The images were saved after swapping channels, and then sorted into categories based on their labels. The resulting dataset for the smart contract is illustrated in Fig. 4.

3.3 ECA-EfficientNet

In this paper, we propose a new network structure called ECA-EfficientNet, which combines the strengths of EfficientNetB0 and EfficientNetV2-s. The main framework of the network is based on EfficientNetB0 with three modifications, as shown in Fig. 5.

Firstly, we used Fused-MBConv from EfficientNetV2 instead of MBConv in the shallow network, which increased the training speed of the network. We also reduced the number of convolutional layers to further speed up training.

Secondly, we used the GWAP method in the ECA module to replace the original Global Average Pooling (GAP) and replaced the SE (Squeeze-and-Excitation) module in the MBConv and Fused-MBConv with an advanced ECA module, which avoids dimensionality reduction on channel attention.

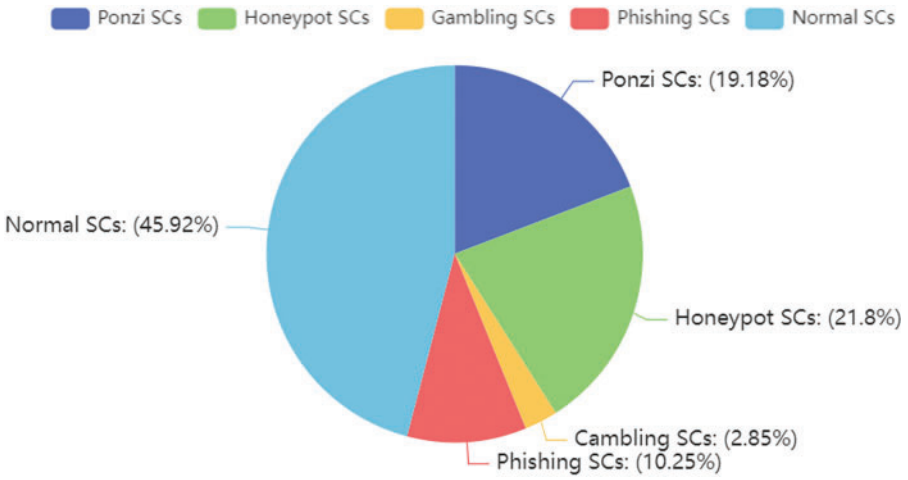

Figure 4: Fraudulent smart contract datasets as a percentage

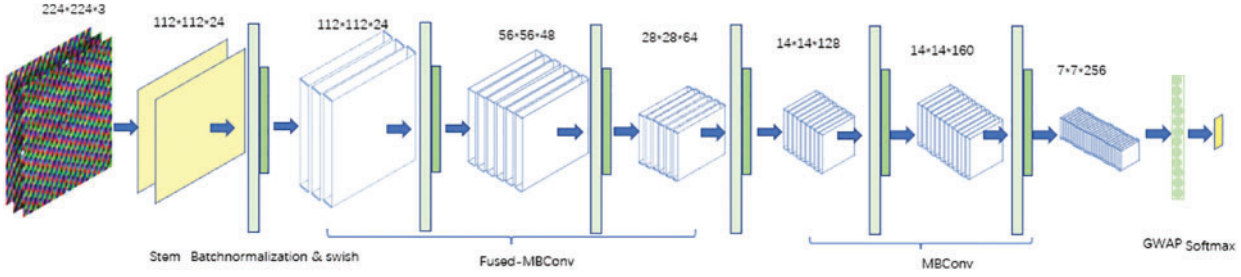

Figure 5: ECA-EfficientNet for identifying fraud smart contracts

Thirdly, we used GWAP instead of a fully connected layer for classification and applied it to each channel feature as GAP. This helped improve the generalization of the model by scaling down to a single value and adding learnable weights to it.

Overall, ECA-EfficientNet is a highly efficient network structure that can achieve better performance than previous models with similar computational costs.

3.3.1 Stem & Swish

As the first part of EfficientNet, the Stem is used for initial feature extraction. Swish is then used for normalization after each convolution, and the Swish [25] activation function is employed to accelerate convergence and prevent gradient disappearance or explosion. Please check for any further grammatical errors and polish the language as necessary.

$$f(x) = x \cdot sigmoid(\beta x) \tag{1}$$

In this paper, the hyperparameter $\beta = 1$ is set to make Swish exhibit ReLU-like properties, as in EfficientNet. Additionally, the language could be further polished depending on the context and intended audience.

3.3.2 MBConv & Fused-MBConv

MBConv is the InvertedResidualBlock used in MobileNetV3 [26], which employs Swish as the activation function. Fused-MBConv, proposed by EfficientNetV2, is used to enhance computing efficiency in the shallow networks. Compared to EfficientNet [27], the structure of EfficientNetV2 can better utilize mobile and server-side accelerators. Additionally, it replaces the 3×3 deep convolution and 1×1 convolution in MBConv with conventional 3×3 convolutions, which fundamentally improves computation speed, as illustrated in Fig. 6.

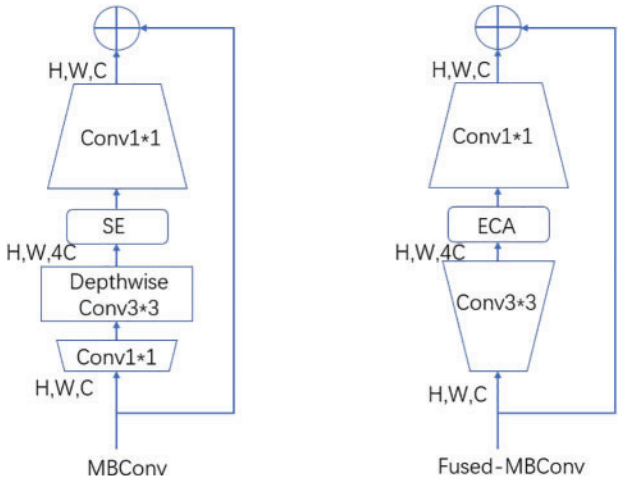

Figure 6: Structure of MBConv and Fused-MBConv

3.3.3 ECA

In this paper, we used the ECA module to replace the SE module [28] in the original network in an attempt to determine the importance of each channel.

The SE module is compressed and downscaled first, and then upscaled, which will result in the loss of the channel importance. By contrast, the ECA module [6] used a local cross-channel interaction strategy without downscaling and is implemented using a one-dimensional convolution with k convolution kernels. We also introduce GWAP to replace the original GAP in the ECA module. The structure of the ECA module in this paper is shown in Fig. 7.

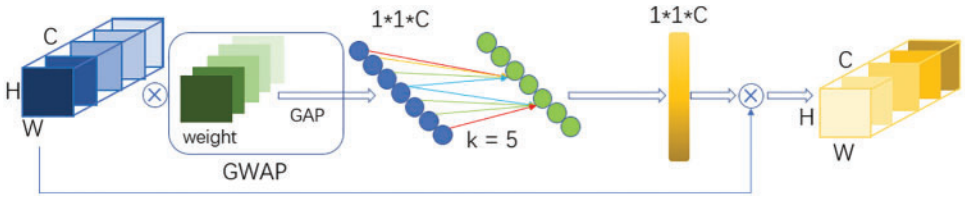

Figure 7: Structure of the ECA after improvement using GWAP

The k value hereby represents the adaptive selection of kernel_size, which is calculated as follows:

$$C = \phi(k) = 2^{(\gamma * k - b)} \tag{2}$$

$$k = \psi(C) = \left| \frac{\log_2(c)}{\gamma} + \frac{b}{\gamma} \right|_{odd} \tag{3}$$

The original paper applied the simplest linear function ϕ (k) $= \gamma * k - b$, combined with the channel dimension C (the number of filters), and set the parameters of the linear function to $r = 2$, $b = 1$. Given the channel dimension C, the kernel size k can be calculated by function (3). $|x|_{odd}$ hereby represents the selection of the nearest odd number. By mapping ψ, the high-dimensional channels have longer interaction distances, while the low-dimensional channels have shorter interaction distances by nonlinear mapping.

3.3.4 GWAP

Instead of using GAP [29] to receive input, the network output employs GWAP. GWAP compresses the features in each channel into a single value, and the weight layer adaptively attaches weights to the features in each layer, as shown in Fig. 8.

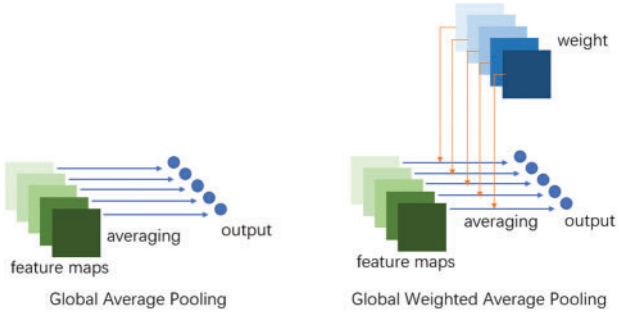

Figure 8: Structure of the GAP and GWAP

In this paper, we propose an adaptive weight layer that is employed to obtain optimal weights through neural network training. GWAP is highly adaptable to EfficientNet, which can effectively improve the model's generalization. Softmax is then used to perform classification.

4 Experiments

In this chapter, we first elaborate on the processing of the dataset used in the experiment. Next, we introduce the experimental parameters and the evaluation metrics selected. Finally, we compare the results of the model training with those of other models and papers.

4.1 Dataset

After collating the data processed in the previous experiments, the resulting datasets consisted of 1494 samples of Ponzi SCs, 1698 samples of Honeypot SCs, 222 samples of Gambling SCs, and 798 samples of Phishing SCs, as well as 1499 samples of Normal SCs after undersampling. The dataset after splitting is presented in Table 1, which illustrates the usage of two datasets in the experiment. The Fruad_dataset is a self-built multi-classification dataset, with a training-to-testing ratio of 3:1, as specified in the Table 1. The Ponzi_dataset is a publicly available Ponzi dataset, and to ensure the

reliability of the model results on real data, the testing set in this dataset consists of 100 unprocessed Ponzi SCs and 100 unprocessed Normal SCs, as indicated in the Table 1.

Table 1: Ponzi_dataset and Fraud_dataset

Dataset	Train_dataset		Test_dataset	
	Smart contracts	Numbers	Smart contracts	Numbers
Fraud_dataset	Ponzi	1120	Ponzi	374
	Honeypot	1273	Honeypot	425
	Gambling	166	Gambling	56
	Phishing	598	Phishing	200
	Normal	1124	Normal	375
Ponzi_dataset	Ponzi	894	Ponzi	100
	Normal	894	Normal	100

After enhancing the data, there still exists an imbalance between positive and negative samples. In dichotomous classification experiments, Normal SCs can be undersampled to match the number of fraud samples, so that the trained model does not overly focus on any one class. However, when performing multiple classifications on the Fraud_dataset dataset, it is inevitable that the amount of data will vary across samples. After expanding the dataset with channel swapping, it is necessary to balance the weights across classes during training. Directly using undersampling to reduce all samples to Gambling SCs with the minimum number of samples would render some data unusable and weaken the effectiveness of the model. Therefore, we added the Class_weight function during training to weight the calculation by comparing the number of fraud contracts in each category.

4.2 Evaluation

To facilitate comparison with experiments in other papers, this paper's experimental evaluation metrics also utilize commonly used classification model metrics such as Accuracy, Precision, Recall, and F1-score. The specific formulas for these metrics are as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

TP (True Positive) indicates the number of positive cases predicted correctly. TN (True Negative) shows the number of negative cases predicted correctly. FP (False Positive) reflects the number of positive cases predicted incorrectly. FN (False Negative) presents the number of positive cases predicted incorrectly.

4.3 Experimental Results

Initially, the ECA-EfficientNet network was trained and evaluated on the self-built Fraud dataset using the ECA-EfficientNet network. This experiment was a multi-classification task. Parameters such as dropout were set according to the original EfficientNet parameters, while other parameters such as batch size were determined using a greedy algorithm, selecting a variable while keeping other variables fixed. To balance the classes before training, the "class_weight" function provided by the Keras framework was used to perform weighting calculations based on the dataset labels for the multi-classification problem.

We implemented each baseline model of EfficientNet based on the parameter settings in papers [25] and trained them on the Ponzi dataset and Fraud dataset. We found from the Ponzi dataset that EfficientNetV2-s has the highest precision and F1-score, but its Recall is lower than EfficientNetB0. Recall is significant for detecting fraud in SCs because it indicates the number of detected fraudulent SCs. Therefore, we decided to improve upon EfficientNetB0 and use EfficientNetV2 as a reference to modify the network structure. As shown in Table 2.

Table 2: Comparison of the performance of each EfficientNet baseline model on the Fraud_dataset and Ponzi_datase datasets

Model	Fraud_dataset				Ponzi_datase			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
EfficientNetB0	0.949	0.945	0.975	0.945	0.978	0.945	0.975	0.945
EfficientNetB1	0.815	0.812	0.799	0.818	0.920	0.812	0.799	0.818
EfficientNetB2	0.884	0.884	0.870	0.885	0.924	0.884	0.870	0.885
EfficientNetB3	0.964	0.959	0.975	0.959	0.909	0.959	0.975	0.959
EfficientNetB4	0.964	0.942	0.955	0.942	0.894	0.942	0.955	0.942
EfficientNetB5	0.891	0.886	0.895	0.886	0.907	0.886	0.895	0.886
EfficientNetB6	0.883	0.876	0.885	0.876	0.867	0.876	0.885	0.876
EfficientNetB7	0.751	0.751	0.835	0.751	0.901	0.751	0.835	0.751
EfficientNetV2-s	0.971	0.972	0.971	0.977	0.932	0.972	0.971	0.977
EfficientNetV2-m	0.926	0.928	0.950	0.924	0.891	0.928	0.950	0.924
EfficientNetV2-l	0.982	0.982	0.970	0.983	0.867	0.982	0.970	0.983

We have primarily modified the original EfficientNetB0 network. Firstly, we have employed Multilayer Perceptron (MLP), Spatial Pyramid Pooling (SPP)+MLP, GAP, and GWAP as classifiers. From the Table 3 above, it is evident that GWAP is the most effective classifier for this dataset. The Accuracy, Recall, and F1-score of GWAP on the same EfficientNetB0+ECA model are 0.973, 0.971, and 0.975, respectively, which are 0.4%, 1.1%, and 0.7% higher than the other classifiers. The use of ECA with the same GAP as the classifier has improved the Precision and F1-score by 2.3%

in EfficientNetB0. The final improved ECA-EfficientNet model has achieved the highest Accuracy, Recall, and F1-score in the experiment. As shown in Table 3.

Table 3: Performance of different improved models in Fraud-dataset

Model	Classifier	Accuracy	Precision	Recall	F1-score
ECA-EfficientNet	GWAP	0.982	0.982	0.982	0.982
EfficientNetB0+ECA	GWAP	0.972	0.973	0.971	0.975
EfficientNetB0+ECA	GAP	0.968	0.968	0.967	0.968
EfficientNetB0	SPP+MLP	0.943	0.943	0.943	0.943
EfficientNetB0	MLP	0.921	0.921	0.921	0.921
EfficientNetB0	GAP	0.949	0.945	0.975	0.945

Based on the experimental results, we speculate that the adaptive parameters of the weight layer in GWAP's improvement of the ECA module considerably enhanced the accuracy of model learning. To prove the advantages of the improved ECA-EfficientNet for fraud smart contract classification, we trained VGG16, ResNet50, and InceptionV3 on the self-built Fraud dataset and obtained the comparison results as shown in Table 4.

Table 4: Comparison of ECA-EfficientNet and various classical classification models in Fraud_dataset performance

Model	Accuracy	Precision	Recall	F1-score
ECA-EfficientNet	0.982	0.982	0.982	0.982
EfficientNetB0	0.949	0.945	0.975	0.945
EfficientNetV2-s	0.932	0.972	0.971	0.977
VGG16	0.951	0.958	0.945	0.951
ResNet50	0.878	0.882	0.873	0.877
InceptionV3	0.865	0.869	0.863	0.866
EfficientNetV2-l	0.982	0.982	0.970	0.983
MobileNet	0.848	0.925	0.735	0.818
RNN	0.797	0.906	0.493	0.681
LSTM	0.812	0.838	0.762	0.798
HTM	0.816	0.875	0.766	0.815

In Table 4, we can see that the ECA-EfficientNet boasts the highest accuracy and recall in comparison to the Fraud dataset. The experiments show that the simplified EfficientNet network not only improves its speed, but also the accuracy is raised from 0.945 to 0.982. Besides, our recall is also increased by 1.2% compared with the largest EfficientNetV2-l. The model can evidently reduce the miss detection rate in the detection of fraud class contracts. Recurrent Neural Network (RNN),

Long Short-Term Memory (LSTM), and Hierarchical Temporal Memory (HTM) are temporal-based network models built with frameworks such as Keras and htm.core. RNN employ a double-stacked architecture, while LSTM use single-layer structure. For the HTM model, images arrays were converted to Sparse Distributed Representation (SDR) data before training.

In Ponzi dataset, the ECA-EfficientNet network is used in the Ponzi SCs dataset after data enhancement as the training set, and the test set remains fixed. Since Ponzi dataset is taken from publicly available datasets, we compared the specific results of F1-score, Recall, etc., with the experimental results in other papers as shown in Table 5 below.

Table 5: Comparison of ECA-EfficientNet performance in Ponzi_dataset with other papers

Model	Accuracy	Precision	Recall	F1-score
ECA-EfficientNet	0.991	0.991	0.991	0.991
SCSGuard [30]	0.922	0.963	0.978	0.971
GCN [4]	Null	0.916	0.916	0.916
CNN [23]	0.973	0.982	0.938	0.959
MTCformer [31]	Null	0.970	0.830	0.890
Se-CapsNet [25]	0.989	0.977	0.989	0.983
ResNet	0.964	0.974	0.946	0.960
VGGNet	0.982	0.978	0.966	0.972
MobileNet	0.944	0.969	0.920	0.943
RNN	0.873	0.942	0.559	0.709
LSTM	0.938	0.970	0.899	0.938
HTM	0.890	0.950	0.852	0.892
LightGBM	0.928	0.950	0.861	0.895
RandomForest	0.934	0.964	0.767	0.854
XGBoost	0.977	0.995	0.920	0.956

In this publicly available Ponzi dataset, ECA-EfficientNet performs the best in terms of all evaluation metrics compared to other papers that conducted experiments on the same dataset. Specifically, it has a precision of 0.982, a recall of 0.982, and an F1 value of 0.982. Compared to the latest paper from MTCformer performs 2.1%, 16.1%, and 10.2% better in terms of precision, recall, and F1 value, respectively. The experimental results show that ECA-EfficientNet has a powerful ability to automatically learn structural and semantic features from smart contract code. As shown in Table 5.

5 Conclusion

In this paper, we utilize only smart contract bytecode to detect known mainstream fraud smart contract types. ECA-EfficientNet achieves the best results on multi-classification and binary classification before the deadline. The bytecode information required by the method is available on the public platform Etherscan, making it more applicable than other methods. Additionally, the channel exchange approach partially solves the problem of extreme data imbalance. However, this method has two drawbacks. Firstly, it is only suitable for Ethereum, and its detection effectiveness for multi-platform is unknown. Secondly, in terms of treating data imbalance, the normal smart contract after

downsampling cannot represent the characteristics of all normal SCs. There are two main research directions for the future. One is to detect multi-platform fraud contracts, and the other is to attempt to solve the extreme data imbalance.

Acknowledgement: None.

Funding Statement: This research work is supported by the National Natural Science Foundation of China, Grant Number: U1603115, Science and Technology Project of Autonomous Region, Grant Number: 2020A02001-1, and Research on Short-Term and Impending Precipitation Prediction Model and Accuracy Evaluation in Northern Xinjiang Based on Deep Learning, Grant Number: 2021D01C080.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: X. Zhou, W. Yang; data collection: W. Yang; analysis and interpretation of results: X. Zhou, W. Yang, L. Wang; draft manuscript preparation: W. Yang, L. Wang. All authors reviewed the results and approved the final version of the manuscript. F. Wei, K. HaiLaTi, and Y. Liao provided valuable input during the preparation of the manuscript and contributed to the interpretation of the results.

Availability of Data and Materials: The data used in this study are available upon reasonable request from the corresponding author, W. Yang. Some data that are proprietary or confidential may be restricted due to legal or ethical reasons.

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

References

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2013. [Online]. Available: http://gavwood.com/paper.pdf

[2] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen *et al.,* "Who are the phishers? Phishing scam detection on Ethereum via network embedding," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Cham, Switzerland, vol. 52, no. 2, pp. 1156–1166, 2020.

[3] R. Agarwal, T. Thapliyal and S. K. Shukla, "Vulnerability and transaction behavior based detection of malicious smart contracts," in *Proc. of CSS*, Cham, Switzerland, pp. 79–96, 2021.

[4] S. Yu, J. Jin and Y. Xie, "Ponzi scheme detection in Ethereum transaction network," in *Proc. of Blockchain and Trustworthy Systems: Third Int. Conf., BlockSys 2021*, Guangzhou, China, pp. 175–186, 2021.

[5] G. Lin, S. Wen, Q. L. Han, J. Zhang and Y. Xiang, "Software vulnerability detection using deep neural networks: A survey," *IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.

[6] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo *et al.,* "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *Proc. of CVF*, Seattle, WA, USA, pp. 11534–11542, 2020.

[7] A. S. Almasoud, F. K. Hussain and O. K. Hussain, "Smart contracts for blockchain-based reputation systems: A systematic literature review," *Journal of Network and Computer Applications*, vol. 170, pp. 102814, 2020.

[8] E. Jung, M. Le Tilly, A. Gehani and Y. Ge, "Data mining-based Ethereum fraud detection," in *Proc. of 2019 IEEE Int. Conf. on Blockchain (Blockchain)*, Atlanta, GA, USA, pp. 266–273, 2019.

[9] L. Liu, W. Tsai, M. Bhuiyan, H. Peng and M. Liu, "Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum," *Future Generation Computer Systems*, vol. 128, pp. 158–166, 2019.

[10] L. Luu, D. H. Chu, H. Olickel, P. Saxena and A. Hobor, "Making smart contracts smarter," in *Proc. of SIGSAC*, Vienna, Austria, pp. 254–269, 2016.

[11] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli *et al.,* "Securify: Practical security analysis of smart contracts," in *Proc. of SIGSAC*, Toronto, Canada, pp. 67–82, 2018.
[12] R. Agarwal, T. Thapliyal and S. K. Shukla, "Vulnerability and transaction behavior based detection of malicious smart contracts," arXiv:2106.13422, 2021.
[13] W. Chen, X. Guo, Z. Chen, Z. Zheng, Y. Lu *et al.,* "Honeypot contract risk warning on Ethereum smart contracts," in *Proc. of IEEE Int. Conf. on Joint Cloud Computing*, Oxford, UK, pp. 1–8, 2020.
[14] C. F. Torres, M. Steichen and R. State, "The art of the scam: Demystifying honeypots in Ethereum smart contracts," in *Proc. of USENIX Security*, Santa Clara, CA, USA, pp. 1591–1607, 2019.
[15] W. Chen, Z. Zheng, E. Ngai, P. Zheng and Y. Zhou, "Exploiting blockchain data to detect smart Ponzi schemes on Ethereum," *IEEE Access*, vol. 7, pp. 37575–37586, 2019.
[16] E. N. Osegi and E. F. Jumbo, "Comparative analysis of credit card fraud detection in simulated annealing trained artificial neural network and hierarchical temporal memory," *Machine Learning with Applications*, vol. 6, pp. 100080, 2021.
[17] E. N. Osegi, "Using the hierarchical temporal memory spatial pooler for short-term forecasting of electrical load time series," *Applied Computing and Informatics*, vol. 17, pp. 264–278, 2021.
[18] A. Kartel, E. Novikova and A. Volosiuk, "Analysis of visualization techniques for malware detection," in *Proc. of EIConRus*, St. Petersburg and Moscow, Russia, pp. 337–340, 2020.
[19] L. Nataraj, S. Karthikeyan, G. Jacob and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. of the 8th Int. Symp. on Visualization for Cyber Security*, New York, NY, USA, pp. 1–7, 2011.
[20] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang *et al.,* "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.
[21] Z. Cui, L. Du, P. Wang, X. Cai and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *Journal of Parallel and Distributed Computing*, vol. 129, pp. 50–58, 2019.
[22] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas *et al.,* "Identification of malicious code variants based on image visualization," *Computers & Electrical Engineering*, vol. 76, pp. 225–237, 2019.
[23] Y. Lou, Y. Zhang and S. Chen, "Ponzi contracts detection based on improved convolutional neural network," in *Proc. of SCC*, Beijing, China, pp. 353–360, 2020.
[24] L. Bian, L. Zhang, K. Zhao, H. Wang and S. Gong, "Image-based scam detection method using an attention capsule network," *IEEE Access*, vol. 9, pp. 33654–33665, 2021.
[25] A. Nader and D. Azar, "Searching for activation functions using a self-adaptive evolutionary algorithm," in *Proc. of the 2020 Genetic and Evolutionary Computation Conf. Companion*, New York, USA, pp. 145–146, 2020.
[26] N. A. Mohamed, M. A. Zulkifley and S. R. Abdani, "Spatial pyramid pooling with atrous convolutional for MobileNet," in *Proc. of SCOReD*, Batu Pahat, Malaysia, pp. 333–336, 2020.
[27] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," arXiv:1905.11946, 2020.
[28] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks," in *Proc. of CVPR*, Salt Lake City, UT, USA, pp. 7132–7141, 2018.
[29] T. Y. Hsiao, Y. C. Chang and C. T. Chiu, "Filter-based deep-compression with global average pooling for convolutional networks," in *Proc. of SiPS*, Cape Town, South Africa, pp. 247–251, 2018.
[30] H. Hu, Q. Bai and Y. Xu, "SCSGuard: Deep scam detection for Ethereum smart contracts," in *Proc. of INFOCOM WKSHPS*, New York, NY, USA, pp. 1–6, 2022.
[31] Y. Chen, H. Dai, X. Yu, W. Hu and Z. Xie, "Improving Ponzi scheme contract detection using multi-channel TextCNN and transformer," *Sensors*, vol. 21, no. 19, pp. 6417, 2021.