applied sciences

MDPI

Article

Modified Genetic Algorithm with Deep Learning for Fraud Transactions of Ethereum Smart Contract

Rabia Musheer Aziz [1], Rajul Mahto [1], Kartik Goel [1], Aryan Das [1], Pavan Kumar [1,*] and Akash Saxena [2]

[1] School of Advanced Science and Languages, VIT Bhopal University, Kothrikalan, Sehore 466116, India
[2] School of Computing Science and Engineering, VIT Bhopal University, Kothrikalan, Sehore 466116, India
* Correspondence: pavankmathsor@gmail.com

Abstract: Recently, the Ethereum smart contracts have seen a surge in interest from the scientific community and new commercial uses. However, as online trade expands, other fraudulent practices—including phishing, bribery, and money laundering—emerge as significant challenges to trade security. This study is useful for reliably detecting fraudulent transactions; this work developed a deep learning model using a unique metaheuristic optimization strategy. The new optimization method to overcome the challenges, Optimized Genetic Algorithm-Cuckoo Search (GA-CS), is combined with deep learning. In this research, a Genetic Algorithm (GA) is used in the phase of exploration in the Cuckoo Search (CS) technique to address a deficiency in CS. A comprehensive experiment was conducted to appraise the efficiency and performance of the suggested strategies compared with those of various popular techniques, such as k-nearest neighbors (KNN), logistic regression (LR), multi-layer perceptron (MLP), XGBoost, light gradient boosting machine (LGBM), random forest (RF), and support vector classification (SVC), in terms of restricted features and we compared their performance and efficiency metrics to the suggested approach in detecting fraudulent behavior on Ethereum. The suggested technique and SVC models outperform the rest of the models, with the highest accuracy, while deep learning with the proposed optimization strategy outperforms the RF model, with slightly higher performance of 99.71% versus 98.33%.

Keywords: Ethereum fraud; logistic regression (LR); k-nearest neighbors (KNN); random forest (RF); support vector classification (SVC)

Citation: Aziz, R.M.; Mahto, R.; Goel, K.; Das, A.; Kumar, P.; Saxena, A. Modified Genetic Algorithm with Deep Learning for Fraud Transactions of Ethereum Smart Contract. *Appl. Sci.* **2023**, *13*, 697. https://doi.org/10.3390/app13020697

Academic Editor: Giancarlo Mauri

Received: 23 November 2022
Revised: 30 December 2022
Accepted: 30 December 2022
Published: 4 January 2023

1. Introduction

Ethereum is a decentralized blockchain platform and famous cryptocurrency alternate platform, in addition to the most well-known platform for peer-to-peer programming. The security of blockchain technology and supervision has been recently receiving much attention. This open-source blockchain technology may be used to create smart contracts. Its development resolves the issue of Bitcoin's restricted scalability. Because of the versatility of the Solidity language, developers may create general-purpose smart contracts. Among the various smart contracts, some may be fraudulent, taking ether from network participants. Ethereum has risen rapidly because of its simplicity and versatility. Ethereum is currently the second most valuable cryptocurrency, only after Bitcoin. Bitcoin and Ethereum have gained in popularity as a result of their creation by Clohessy et al. [1], Li and Whinston [2], and Liu and Serlaetis [3]. The major purpose is to empower individuals by allowing employers to manage their own data and transactions. One such cryptocurrency is Ethereum, which was established by Vitalik Buterin. In brief, Ethereum is a cryptocurrency allocation system that allows a user to send fractional cryptocurrency to anybody at a low cost; because Ethereum is built on a blockchain network, everyone may use the digital transactions, with extremely minimal transaction fees, in a secure manner. The vast user base of Ethereum encourages developers to release their programs on the network, further solidifying Ethereum as the primary platform for decentralized programs such as DeFi and

NFTs. A much more scalable infrastructure will soon be made available by the backwards-compatible Ethereum 2.0 protocol, allowing developers to create decentralized apps with higher transaction throughput. A smart contract is a part of an existing application that is stored at a particular contract location on the blockchain. Apps can invoke the functionalities of smart contracts, alter their states, and start transactions [4,5]. The Ethereum Virtual Machine (EVM) compiles programming languages such as Solidity and Vyper into bytecode, which is then used to implement smart contracts on the blockchain technology. The growth of artificial intelligence (AI) and various mining techniques has increased the demand to exchange the most diversified data; however, fraudulent activities have also appeared in online services. Unfortunately, the development of these data pertains to numerous distinct security sectors, so sharing them is difficult, especially when they contain information that requires privacy and security. The integration of automation in blockchain has resulted in the fast acceptance of the technology in several industries, such as online finance, the Internet of Things (IoT), supply chain management, healthcare, insurance, and so on [6,7]. Blockchain is a global and immutable ledger that makes it easier to record transactions between both receiving and sending parties and monitor assets in a corporate network. Meanwhile, Ethereum is a P2P network for securely implementing and verifying application code, referred to as smart contract accounts. Smart contracts enable parties to transact with one another even without a trusted centralized government.

There is a concern that criminal transactions may be concealed. However, owing to the nature of the blockchain, all transaction data are public, and anybody may obtain transaction information. Nevertheless, as wallets and transactions for cryptocurrencies grow in quantity and become more common, detection becomes increasingly difficult. The fact that the blockchain is decentralized indicates that no one entity has control over it. If a scam occurs, it would be difficult to identify the perpetrators. These concerns, along with the users' anonymity, may lead to fraudulent activity. The detection of suspicious transactions in such vast financial transaction networks has been the subject of several research works. The majority of these works have modeled fraudulent transaction patterns based on the timestamps and quantities of one or more transactions as features and used these models to detect unknown transactions. Much research has been undertaken to optimize models and evaluate fraud detection. The goal of this study is to enhance the suggested methodology for detecting fraudulent actions on the blockchain. Anomalies in the Ethereum blockchain's transactional data are investigated. Transactions that depart from the norm are considered abnormal or suspicious. Furthermore, whether these transactions are lawful or criminal, they should be investigated. We ran a large experiment to compare several machine learning models using various performance metrics [8,9].

1.1. The Objective of the Paper

- A deep learning model has been developed to predict fraudulent transactions in real time. It can be used to improve the efficiency of the system by implementing a novel optimization technique. The proposed model with large datasets has been taken into account considering that it can keep up with the growth of the Ethereum network dataset and still predict accurately.
- To propose a model that can detect fraudulent activities on the blockchain of Ethereum.
- To propose a deep learning approach with a limited number of features for Ethereum fraud detection.
- An evaluation of the proposed model's performance against different models is performed. It takes into account the various performance measures used by different classifiers.
- To implement the model independently of the user interface.

1.2. Proposed Novel Work

According to the literature study, the majority of earlier work for identifying fraudulent transactions used machine learning algorithms and optimization techniques such as

random forest (RF), decision trees (DT), support vector machine (SVM), and so on. However, there has been relatively little work done using the deep learning technique, which has the ability to provide results very fast and precisely. Furthermore, when compared to random forest and other prominent approaches, deep learning outperforms numerous high-performing algorithms when addressing problems in several disciplines. In order to overcome overfitting, the suggested work uses a deep learning technique based on novel GA-CS optimization. This strategy can accurately identify anomalous transactions. Using the suggested algorithms as the primary algorithms for our research, we will conduct a thorough comparison with other generally used algorithms such as random forest, regression, and so on. Additionally, we tweak the suggested model using hyperparameter tuning to improve its performance. With great efficiency and minimal memory utilization, the proposed model will anticipate Ethereum transaction fraud. Figure 1 displays the workflow of the process followed in this study.

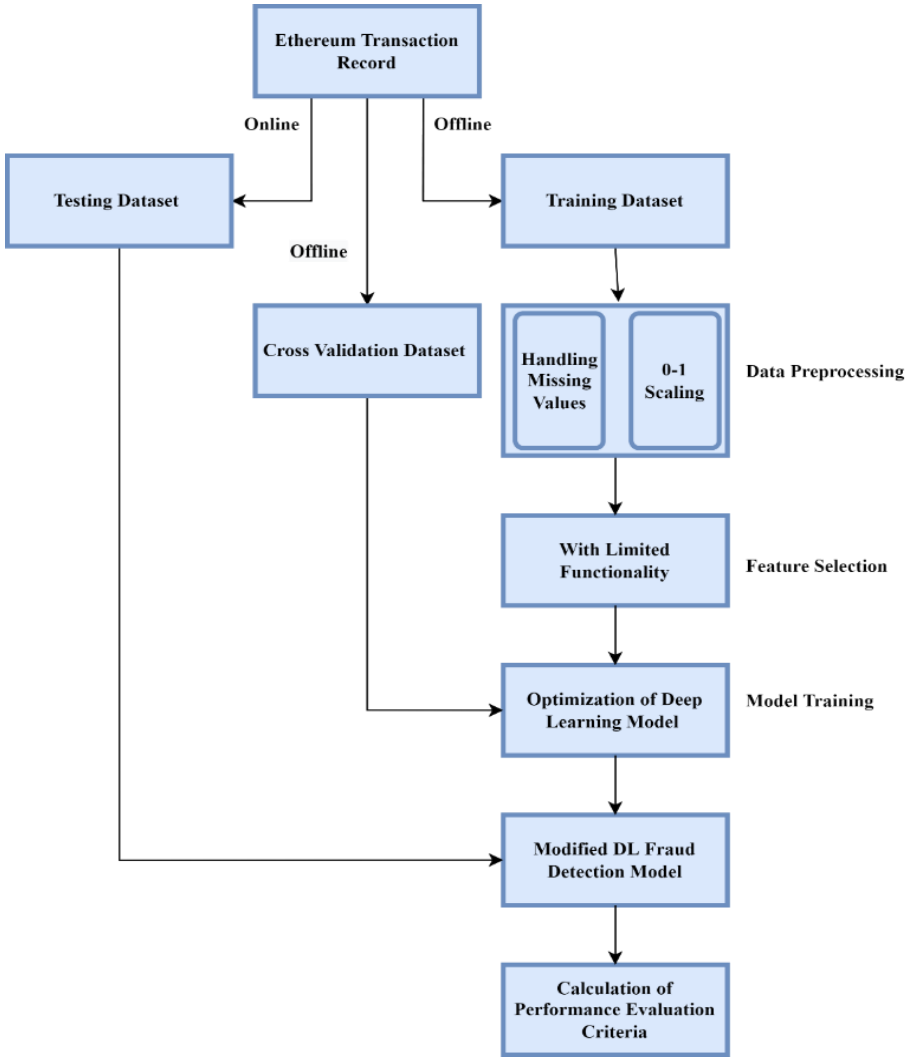


Figure 1. Workflow of the process followed in the study.

1.3. Paper Organization

The remainder of the paper is coordinated as follows. Section 2 contains the literature summary for this study. Section 3 considers categorization modeling and the underlying principles of the proposed method. Section 4 describes the data preparation, experimental setup, and efficiency metrics for several methods. Section 5 presents the experimental data. Section 6 closes with a recap of previous work and numerous ideas for future review.

2. Literature Review

One of the most well-known types of Ethereum fraud is the Ponzi scheme. Jung et al. [10] developed a data mining methodology to identify Ponzi fraud, a system in which only early investors gain from the money collected from later investors and, after a period of time, the plan collapses, leaving later investors with nothing. They constructed their dataset by using open-source Ethereum smart contracts, which included previously identified Ponzi smart contracts, and discovered that their pyramid scheme fraud detection model based on data mining has precision of 0.99 and recall of 0.99 and can protect against these deceitful contracts.

These misleading pyramid schemes were thoroughly investigated by Bartoletti et al. [11], who also evaluated their nature and effects from a variety of angles. By retrieving their Solidity codes and then confirming them with the blockchain EVM codes, they created a dataset of these fraud schemes. The most likely scams, according to a thorough investigation of these schemes, are those that promise large profits, explain their dependability with the justification of open access to the source code, and have a Gini coefficient of more than 80%.

Chen et al. [12] presented a categorization model not relying on the source code to automatically identify fraudulent schemes using smart contracts with labels and extracting essential features without source code. They constructed their dataset by manually checking from etherscan after collecting open-source smart contracts. They concluded that their proposed model can identify fraudulent schemes at the time of their creation, and their suggested method (RF) increases the precision to 0.95 and outperforms other methods such as XGBoost and one-class SVM.

Vasek et al. [13] studied the extensive network of fraudulent pyramid schemes that entice Bitcoin users and looked at the availability and demand of these schemes. By collecting information on victims' dialogues with scammers, they also examined the demand side and supply side of these scams. They collected data by researching the full history of scam allegations, gambling, and games involving investments, as well as information from the locations wherein these fraudulent incidents were marketed. They came to the conclusion that if the scammer maintains contact with the victim, the scam lifetime increases, and if the victim maintains more frequent daily updates, the fraud lifetime falls.

Ajay et al. [14] suggested using secure smart contracts built on the ERC20 interface on a blockchain network with the necessary functions and procedures, to provide a comprehensive framework for safeguarding manufacturing activities based on the Cloud. It is very difficult to uncover features that allow the precise identification of anomalous contracts, and statistical analysis based on these attributes is equally unsuccessful. Second, they disregard the instabilities and internal consistency of smart contract accounts, which largely contributes to model obversion. Ali Aljofey et al. [15] employed a web crawler to collect data and obtained unique bytecodes for contracts confirmed by etherscan.io, which they then translated into opcodes using the pyevmasm disassembler. A feature vector was built by combining transaction characteristics, opcode n-grams, and document frequency. To construct a powerful classifier, the ensemble learning model was used as a feature vector together with the extra trees and gradient boosting machine learning techniques. This entire model was then utilized to categorize anomalous contracts. In contrast to other algorithms, such as DT, XGB, LGBM, and GBM, the results revealed accuracy of 89.67%, demonstrating the effectiveness of the proposed model. Combining characteristics significantly improves detection. IoT devices are more often secured through firmware updates. Traditional updating systems have drawbacks, including bandwidth constraints and attacks by hackers' distributed denial of service (DDoS).

Runnan Tan et al. [16] presented a model that utilizes web crawlers to collect tagged bogus addresses, after which a transaction network is rebuilt using the public transaction book. Then, to extract node attributes for detecting fraudulent transactions, an amount-based network embedding approach is developed. Ultimately, a graph convolutional network is used to differentiate between legal and counterfeit addresses. Results show

accuracy of 95%, which indicates the system's high effectiveness in identifying fraudulent Ethereum transactions.

Qi Yuan et al. [17] proposed a model to discriminate between phishing and non-phishing nodes, and an SVM approach was used. The detection objective was 1259 phishing nodes, and 1259 unlabeled nodes at random as outliers were chosen. To build a subnetwork, the aforementioned nodes' first-order neighbors and transaction records were crawled. After obtaining a massive Ethereum transaction network, nodes were depicted using a network embedding approach. Word-embedding-inspired random-walk-based network embedding approaches, which try to maintain the local structural properties, have been used, and node2vec, an outstanding random-walk-based method, was employed to represent nodes. At this stage, the SVM classifier is used. Results showed that non-embedding approaches are ineffective in detecting phishing. Non-embedding methods are methods for manually extracting characteristics that, unlike embedding methods, cannot use structural information. Therefore, to obtain transaction network characteristics, the network embedding technique is crucial. It has been stated that Ethereum scams generate large profits and represent a severe threat to the Ethereum network's financial security.

R. F. Ibrahim et al. [18] investigated unlawful accounts on the Ethereum blockchain and proposed a fraud detection model with three different algorithms: decision tree, random forest, and k-nearest neighbors (KNN). A dataset was used from Kaggle, with 42 features; later, it was reduced to only six features with the help of correlation coefficients, which selected only the important features. Results demonstrate a considerable improvement in time measurements when utilizing the three algorithms, as well as an improvement in the F measure when using the random forest approach. An effective solution for fraud detection is desperately needed to ensure a secure investment environment. In this research, a three-step methodology for mining Ethereum transaction data to detect phishing frauds is offered.

Woei-Jiunn Tsaur et al. [19] advocated for utilizing a distributed database to minimize storage space by not placing firmware information directly within it. The correctness and integrity of the acquired firmware of IoT devices may be assured after downloading the firmware in the suggested system. Instead of consulting the manufacturer, IoT devices obtain firmware from the distributed database's download point. By utilizing blockchain technology, it is possible to lessen the strain on the file server while keeping system information safe. This research may lessen the requirement for storage space while also improving system security. The suggested solution performs well in several respects, such as firmware integrity, IoT device connection security, security mechanisms, and device anonymity. Due to the properties of this technology, such as immutability and transparency, the technology has been lengthened further than cryptocurrency and is being used in various sectors, such as education, healthcare, finance, energy, government, and IoT, to provide more privacy, increased efficiency, and increased protection.

3. Proposed Model

Deep learning has lately emerged as a popular issue in the field of machine learning. Deep learning techniques include convolutional networks, deep belief networks, and deep autoencoders, which are hierarchical learning structures with several layers of input processing for representation learning or pattern classification [20–22]. Deep learning may be traced back to the study of artificial neural networks. A back-propagation algorithm is a typical method for training neural network weights. However, as the depth of the neural networks increases, the efficacy of the back-propagation process decreases significantly, possibly due to difficulties such as low local optima and error dilution. The fundamental notion of unsupervised layer-by-layer greedy learning is credited with empirically eliminating the optimization issue in deep architecture training parameters. Deep learning (DL) has made considerable strides in recent years in handling many real-world challenges. The success of DL may be credited mostly to its design, the optimization approach utilized, and the tweaking of hyperparameters to recognize various patterns in data [23–25]. This research

attempts to identify an optimization strategy that will play an important role in DL to obtain better outcomes. The major difficulties that a learning algorithm faces are becoming trapped in local minima and a poor pace of convergence. GA-CS, a novel optimization approach, is used in conjunction with DL to address these difficulties. The proposed GA-CS algorithm is a combination of the GA and CS methods that helps to avoid being trapped in local optima and enhance the global search to find better solutions. The method GA-CS is proposed to overcome the shortcomings of CS by introducing a Genetic Algorithm (GA) in the exploration phase of the CS approach. The new optimization technique GA-CS is used with DLANN to solve problems of optimization in DL. The following is a step-by-step overview of the GA-CS algorithm (Algorithm 1):

1. Common cuckoo: It picks host nests in a group having similar egg aspects to its own.
2. Other cuckoos: It picks host nests in a group having dissimilar egg aspects to its own.
3. Some other cuckoo species: Some cuckoos lay cryptic eggs, which are dark in color and do not resemble the host's eggs. Instead, they are hidden in dark domed nests, where it is hoped that the host will not find them. Figure 2 shows the flow chart of the modified cuckoo search.

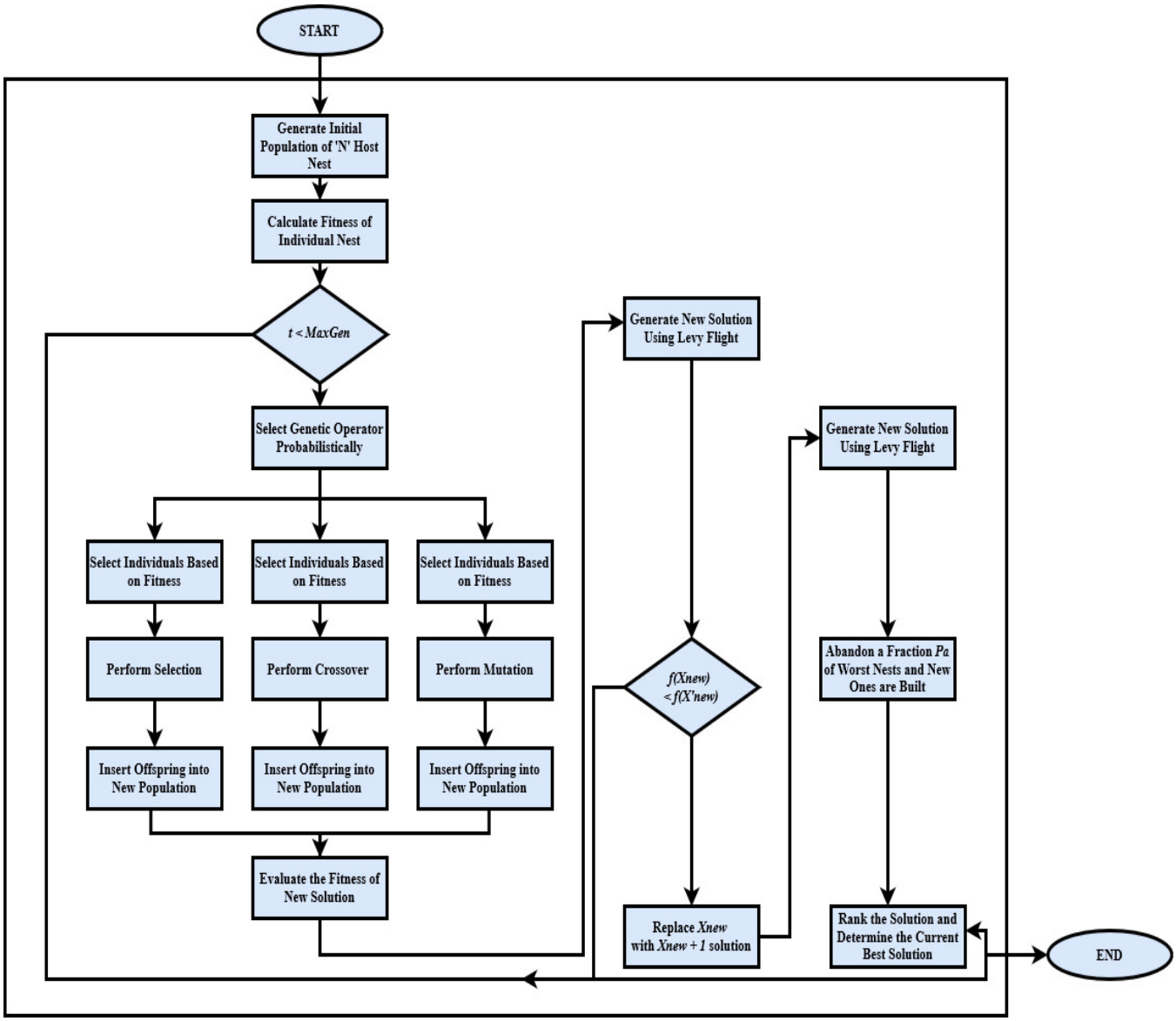


Figure 2. Flow chart of proposed optimization algorithm.

The flowchart of the GA-CS algorithm is shown in Figure 2. Three different types of nests selected by cuckoos for laying their eggs concerning the similarity or dissimilarity of egg characteristics are as follows [26–28].

Algorithm 1. Proposed optimization algorithm (GA-CS).

Pseudocode of GA-CS algorithm:

- Initialize All Choices For The Dataset At The Outset.
- Initialize An Initial Population Of N Host Nest With M Eggs.
- While (X < Generation)
- For Cycle = 1 To Maximum Cycle Number MCN Do
  a. For Every Nest
- Assign Cuckoo Type At Random (Say, *i*)
- Initialize Two Eggs Using Crossover With The Two Eggs In The Nest
- Set The Better One Out Of Two Eggs.
- Else
- Initiate An Egg To A Random Result (Cryptic Egg)
- End If
  i. Compute Quality or Fitness F_i
  ii. Initialize A Nest Among N (Say j) Randomly If (F_i > Fj)
  iii. i^{th} The New Solution
  iv. End If
  v. Sort The Eggs According To The Result.
  vi. In The Nest Assign The Best Response Among M Eggs
  vii. A Partial Part (P_a) Of Substandard Nests Is Deserted And New Are Constructed;
  viii. Keep The Nests With Best Quality Solutions;
  ix. End For
  x. Rank The Solution And Find The Prevailing Best;
  xi. End While
- Note Down The Best Solution So Far.
- Update Cycle = Cycle + 1
- Until Cycle = MCN

The proposed algorithm is used to improve the performance of the DL model.

4. Data Pre-Processing and Experimental Setup

- Data Acquisition

The dataset is available on the Kaggle website (https://www.kaggle.com/code/sukantokumardas/fraud-detection-ethereum-transactions/data (accessed on 15 November 2022)) and was created by compiling the Ethereum classic code (ETC). The dataset contains 9841 transactions or rows that have been identified as either fraudulent or valid Ethereum transactions. As previously stated, the transactions table with 17 fields is the major focus of this research. These fields are used to detect Ethereum network abnormalities using machine learning approaches and algorithms. This dataset was initially referred to by Steven et al. [29].

4.1. Pre-Processing

The dataset is imbalanced, which could skew the model's accuracy. The minority upscale should be resized to meet the periodicity with the majority of the dominant class in order to equalize the classes.

Figure 3 depicts that, in the raw data, 7662 of the 9841 elements has a value of "0". The dataset now contains 2179 records after invalid entries are deleted, since they cannot be construed as suspicious or legitimate transactions, such as transactions with zero values. The pie chart shows that 78% of transactions are fraudulent and the remaining 22% are legitimate.

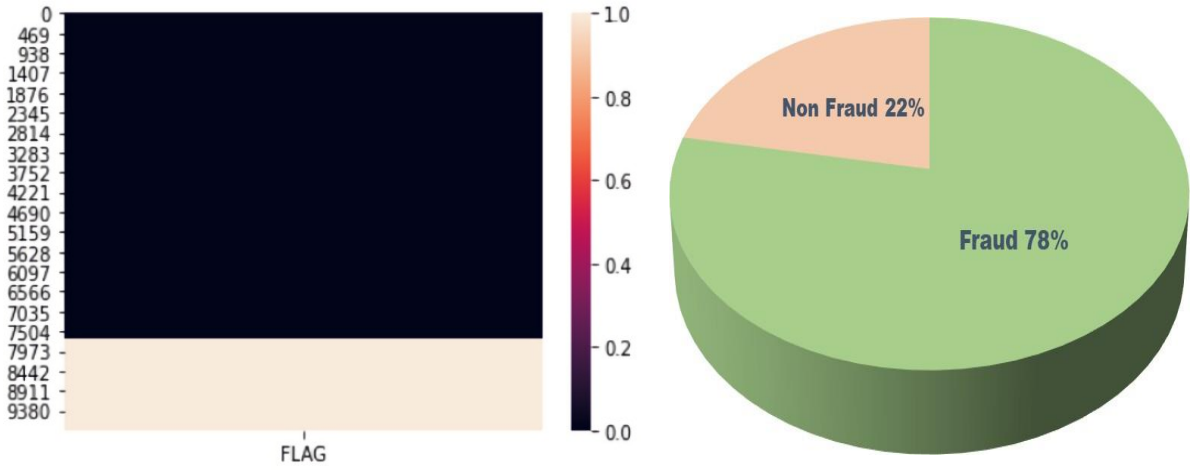


Figure 3. Ethereum classic dataset distribution having '1′ and '0′ was used and fraudulent value percentage.

After calculating the percentage of missing values of columns in rows and visualizing the heatmap, the heatmap in Figure 4 shows that there are several Ethereum transactions with missing attribute values, rendering the data unusable for further processing. Thus, we eliminate the features with the highest number of null values.

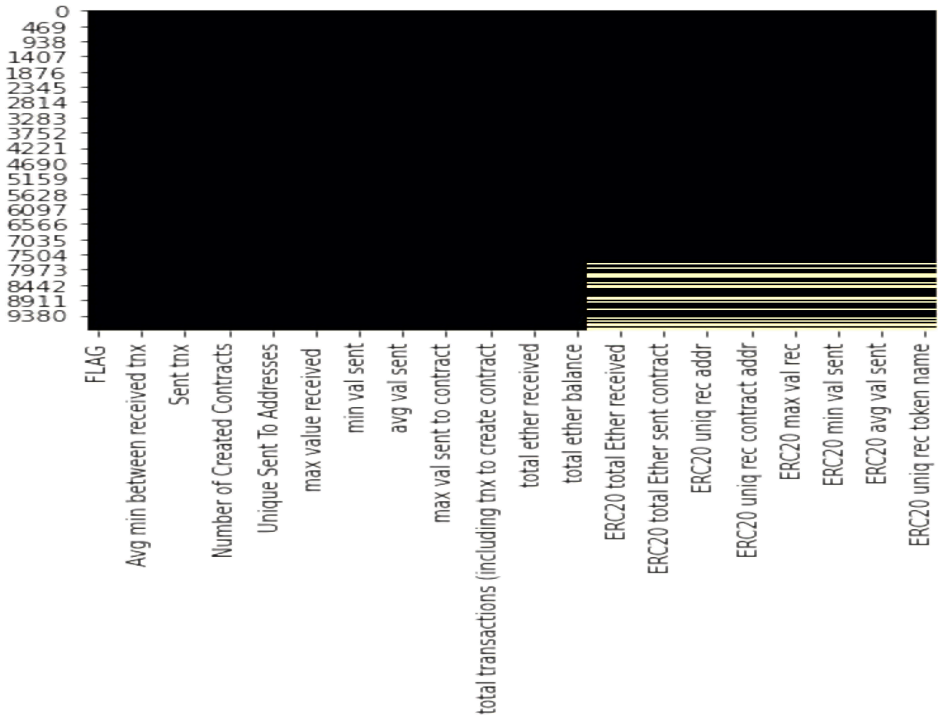


Figure 4. Heatmap showing the dataset's missing values.

4.2. Distribution of Features

The distribution of the attributes is shown in Figure 5. The parameters "ERC20 Uniq sent to addr.1" and "min value sent to contract" both include a number of null values, as can be observed. Such formations will be eliminated as a result since they no longer match the model.

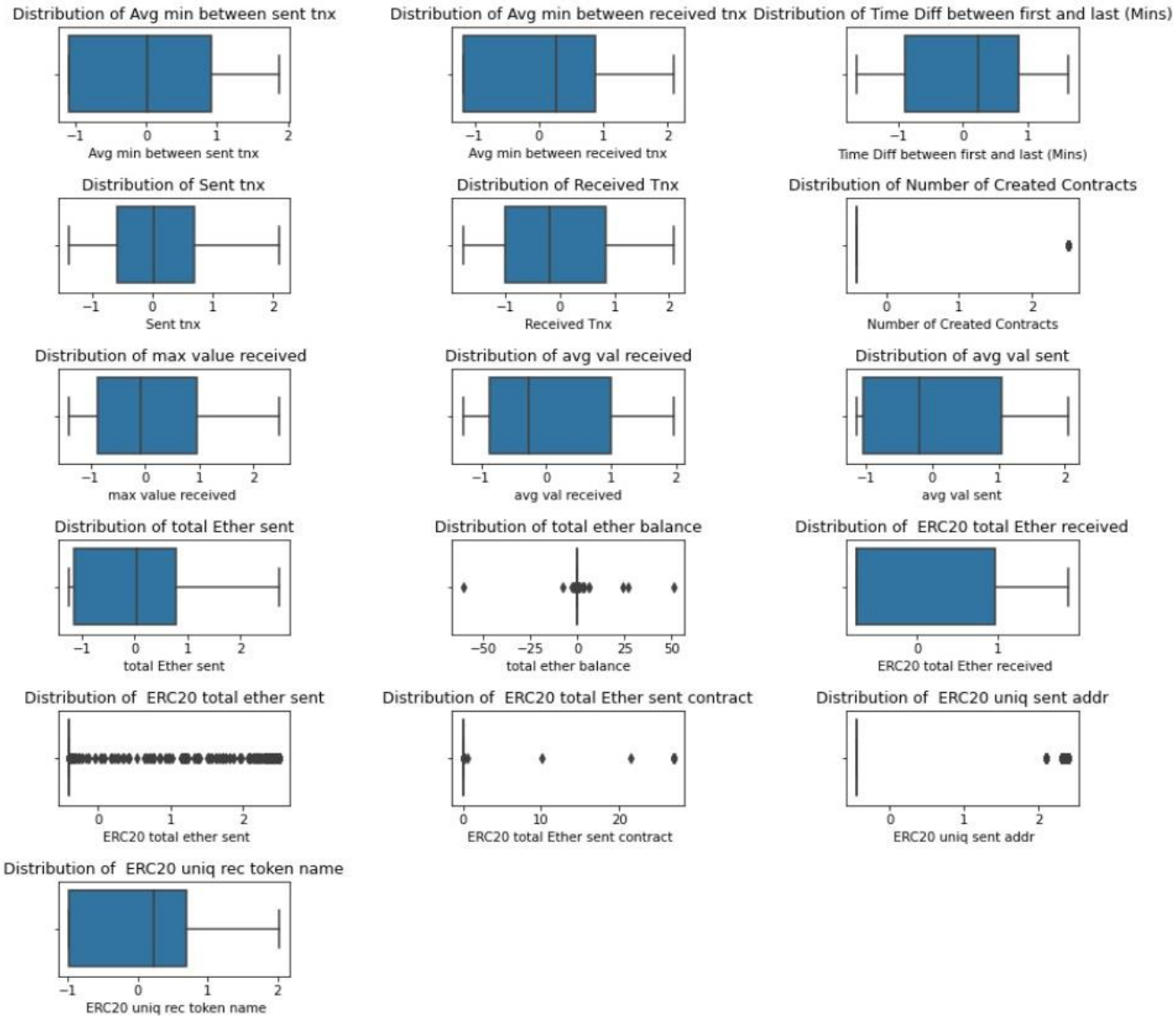


Figure 5. Boxplots depicting the distribution of features post log transformation.

4.3. Dataset Partitioning

The database is split into two parts: a retraining dataset used by the model to adjust to the data, and a testing dataset used to verify and reinforce the developed model's accuracy. The dataset for this study is split into training and testing portions at a ratio of 4:1, or 80% to 20%, respectively.

4.4. Normalization of Training Features

The dispersion of properties after multinomial logistic regression is examined using the power transform function.

4.5. Imbalanced Data Management

Oversampling is used in the classification balancing approach of the machine learning technique known as SMOTE, or the Synthetic Minority Oversampling Methodology. The issue that arises most commonly while training a model is imbalanced classification. Instead of eliminating a large number of entries from the dataset, this strategy repeats values inside the minority class. However, these recurring values do not offer any fresh information.

BEFORE OVERSAMPLING: Fraud: 1757, non-fraud: 6115.

AFTER OVERSAMPLING: Fraud: 6116, non-fraud: 6115.

In Figure 4, boxplots are used to show the distribution of features, which aids in visualizing the spread and skewness of the data, as well as potential outliers. The "whiskers" extending from the box show the minimum and maximum values of the data, and any points outside of the whiskers are considered outliers.

4.6. Proposed DL Architecture

We structured the data as matrix vectors before applying DL on the selected set of features. The data matrix was then analyzed and categorized using the DL model that has been proposed.

In this model (Figure 6), the number of layers used is 8, including 6 convolutional layers, 1 flatten layer, and 1 dense (fully connected) layer. The convolutional layers are used to extract features from the input data, while the fully connected layer is used to make the final classification decision. The convolutional layers are characterized by their kernel size, which is specified as 2×2 in this model, and the number of filters or kernels used, which is specified as 8, 16, 32, 64, and 128 in the first through fifth convolutional layers, respectively. The kernel size determines the size of the receptive field of each convolutional neuron, while the number of filters determines the depth of the layer. Batch normalization is a technique that is used to normalize the activations of a layer over a mini-batch of input data. It helps to improve the stability and generalization of the model by reducing the internal covariate shift, which is the change in the distribution of the activations of a layer as the model is trained. Rectified Linear Unit (Relu) is a non-linear activation function that is widely used in deep learning models. It has the advantage of being simple and computationally efficient, and it has been shown to improve the performance of neural networks on many tasks. Max pooling is a down-sampling operation that is used to reduce the spatial dimensions of the feature maps produced by the convolutional layers. It is typically applied after the convolutional and activation layers and helps to reduce the number of parameters in the model and improve its generalization performance. The model ends with a flatten layer, which is used to flatten the output of the final convolutional layer into a 1D vector. This is followed by a dense (fully connected) layer, which is used to make the final classification decision, and a softmax activation function, which is used to produce probability scores for each class. The output layer then produces the final classification predictions. Consequently, the parameter of output in the fully connected layer denotes the number of classes in the data; for instance, an output size of 2 indicates that there are two classes in the data.

The hybridized GA-CS algorithm, which will be implemented as an optimization tool over the deep learning model proposed above in Figure 6. This algorithm is explained in detail in Section 3, which presents the proposed model.

4.7. Parameter Setting of Proposed Models

Parameter P_a: It indicates the probability of discovering an egg. In the tweaked cuckoo search algorithm, the P_a value is modified dynamically using the following Equation (1).

$$P_a = P_{amax} - \frac{P_{amax} - P_{amin}}{iter_{max}} * iter \quad (1)$$

In Equation (1), P_{amax} denotes the maximum probability of discovering an egg. This is the highest probability that the cuckoo has of finding an egg in a given nest. P_{amin} denotes the minimum probability of discovering an egg. This is the lowest probability that the cuckoo has of finding an egg in a given nest, $iter_{max}$ is the maximum number of iterations that the cuckoo will attempt to find an egg in a given nest. This is the maximum number of times that the cuckoo will search for an egg before giving up and moving on to another nest.

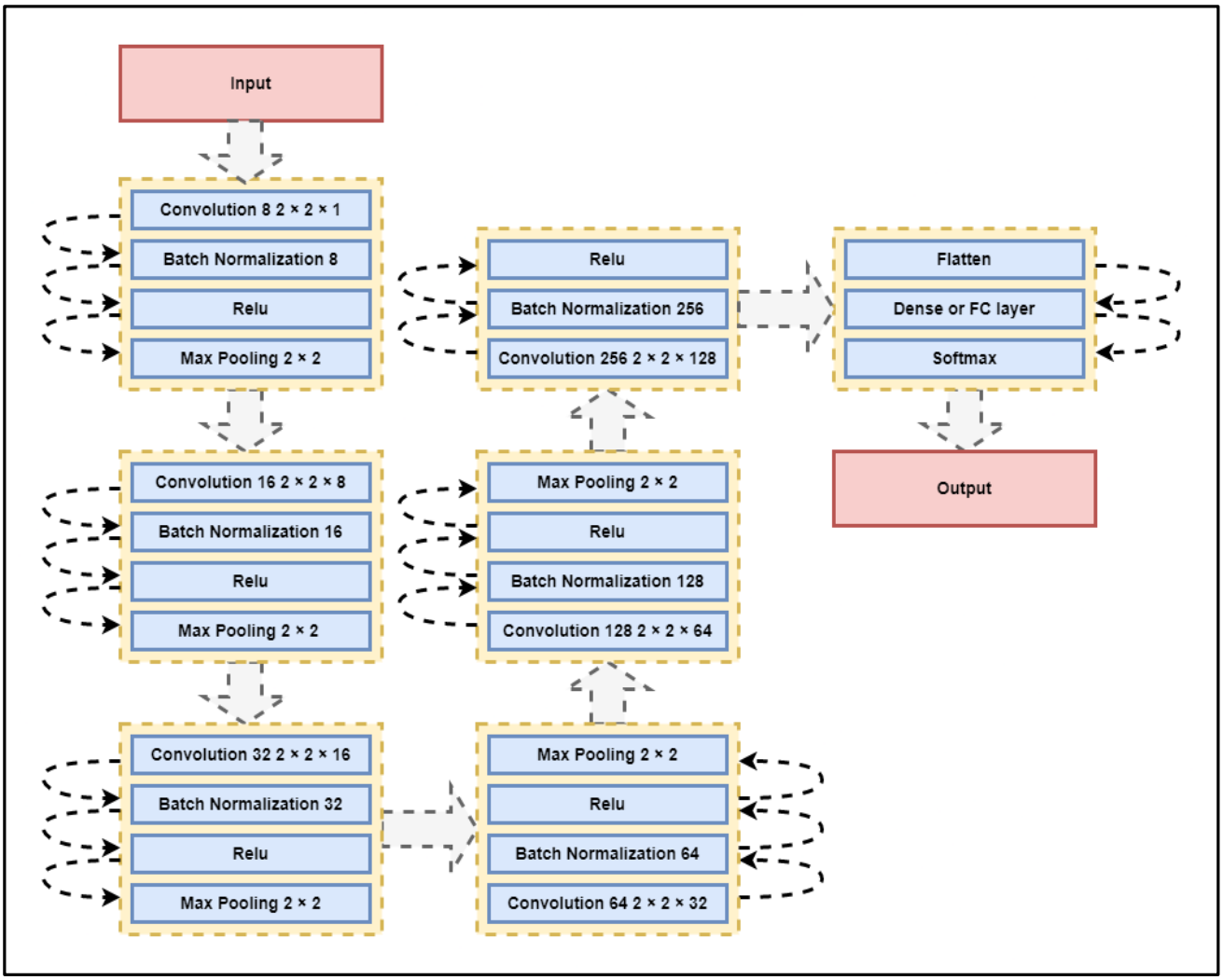


Figure 6. Configuration of deep learning model used in this study.

Fitness function: The following formula is used to determine the proposed model's classification accuracy (Equation (2)):

$$Classification Accuracy = \frac{CC}{N} \times 100 \quad (2)$$

In Equation (2), N is the total number of samples in the relevant class, while CC denotes the correctly categorized observations. This suggested method's fit function is assessed using the classifier's classification accuracy. The prior result is ignored in favor of the present one if the fitness level value is higher than the prior one; else, the prior solution is preserved. Finally, we have Equation (3) below:

$$Fitness\ (f) = Accuracy(f_a) \quad (3)$$

Accuracy (fitness (f)): Analyzing the accuracy of the classifier using the data (f). In this paper, a 10-fold approach is used to find outcomes that were objective. We repeat the experiment with 10-fold validation to determine the ideal answer for the suggested technique. The parameters of the GA-CS algorithm that was used in our experiments are given below.

In Table 1, a list of all relevant parameters for the cuckoo search that were employed in our proposed model is provided. Nest represents a potential solution to the optimization problem, and the algorithm uses a process called "abandoning" to choose which nests to keep and which to replace with new solutions. Total number of eggs specifies the number of cuckoo eggs that will be placed in the nests at each iteration. The number of eggs can be adjusted to balance exploration and exploitation in the search process. Total number of generations specifies the maximum number of iterations that the algorithm will run for. The algorithm will stop once it reaches the specified number of generations. Limit specifies the maximum number of times that an egg can be abandoned in a nest before it is removed. This parameter can be used to control the rate at which new solutions are introduced into the search process. Step size specifies the maximum distance that an egg can move from its current position in the search space at each iteration. The step size can be adjusted to control the rate at which the algorithm explores the search space. Mutation probability specifies the probability that an egg will undergo a mutation at each iteration. The mutation process can be used to introduce new solutions into the search process and improve the diversity of the solutions being considered. Crossover rate specifies the probability that two eggs will undergo crossover, which involves combining their solutions to create a new solution. The crossover process can be used to improve the diversity of the solutions being considered and to introduce new solutions into the search process.

Table 1. Parameter setting of the proposed algorithm.

Parameter	Value
Nests	80
Total no. of eggs	10
Total no. of generations	300
$P_{a_{min}}$	0.2
$P_{a_{max}}$	0.6
Limit	5 iterations
α step size	1.7
Mutation probability	0.03
Crossover rate	0.75

5. Experimental Results and Discussion

The dataset was factionalized into training and testing datasets, each having an 80% to 20% ratio, respectively, with the aid of the Python sklearn module. Selecting the appropriate hyperparameters is crucial while modeling. Only the elements that significantly affect model performance were chosen for parameter adjustment to enhance the real-time fault detection performance. Scores from the proposed framework were compared using the F1 score and precision, and the suggested framework's accuracy was evaluated.

Additionally, Figure 7 shows that the model-based version of the proposed technique outperforms it, without sacrificing other parameters or overfitting, by offering the highest level of accuracy in addition to SVC, following the hyperparameter adjustment.

In Table 2, the outcomes in terms of testing accuracy, training accuracy, F1 score, recall, and precision are displayed. The recall and precision came from the confusion matrix of an alternative model with an alternative classifier. It can be observed that random forest and logistic regression had the lowest accuracy of all the models.

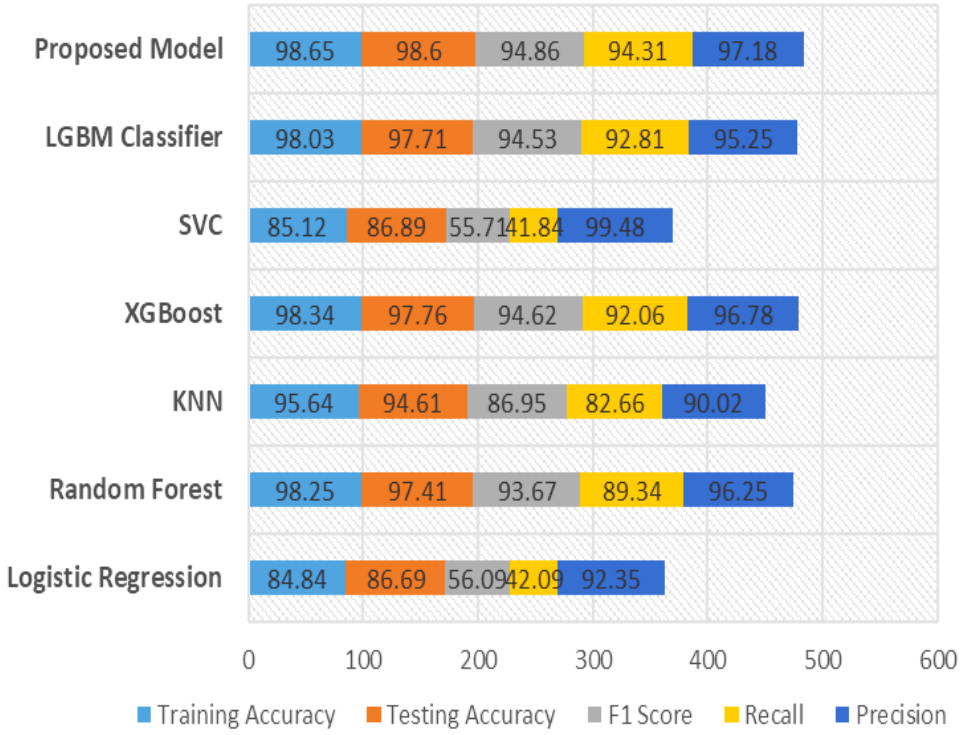


Figure 7. Comparison of scores for the proposed model.

Table 2. Comparison of performance of the proposed algorithm with popular published algorithms.

Type of Algorithm	Training Accuracy	Testing Accuracy	F1 Score	Recall	Precision
Logistic Regression	86.16	88.69	86.16	88.4	92.46
Random Forest	99.57	99.41	93.74	90.65	98.36
MLP Classifier	97.85	97.07	88.39	89.49	92.57
KNN	96.96	96.61	87.02	82.97	90.13
XGBoost	97.66	95.76	94.69	93.37	96.89
LGBM Classifier	96.44	95.89	93.78	92.15	99.59
SVC	99.33	98.33	94.6	93.12	95.36
Proposed Algorithm	99.27	99.71	96.93	94.62	97.29

In Figure 8, the training and testing accuracy and loss are plotted against the number of epochs; this helps us to track the model's performance over time and identify patterns or trends in the model's behavior. The testing accuracy is a measure of how well the model is able to generalize to new, unseen data. It is calculated as the proportion of correct predictions made by the model over the total number of testing examples. High testing accuracy indicates that the model is able to make accurate predictions on new data and is not overfitting to the training data. The training loss is a measure of the model's error on the training data. It is calculated as the average difference between the predicted output and the true output for all training examples. A low training loss indicates that the model is able to accurately predict the output for the training examples. In our model, the training accuracy and loss are consistently high and the testing accuracy and loss are consistently low, which indicates that the model is learning effectively and is not overfitting to the training data.

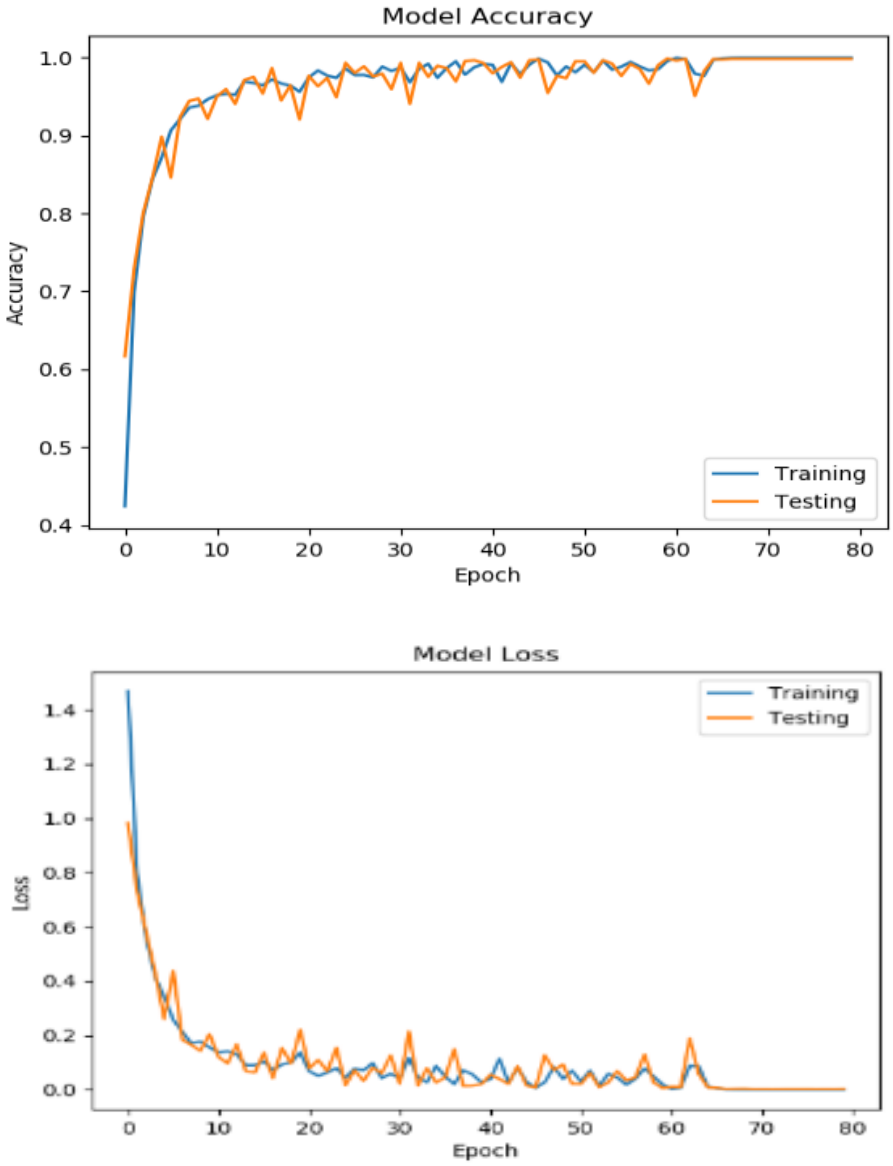


Figure 8. Training and testing accuracy and loss vs. epochs for the proposed model.

5.1. Best Model Evaluation

Following the implementation of the suggested approach using the DL classifier, as displayed in Table 2, we obtained the maximum training and testing accuracy. It regulates a number of model characteristics, including data overfitting and underfitting. Therefore, for the proposed approach, we employed randomized hyperparameter tuning and achieved good accuracy with the chosen parameters. The ideal parameter values for the best result are shown in Table 2.

5.2. Importance of Features

The below Figure 9 depicts a feature comparison graph, which examines the relative value or relevance of several features in a dataset. The importance value of a feature is assessed to determine how much the feature contributes to the prediction or classification accuracy in the ideal model, in order to comprehend the relevance of each element. "Time difference between initial and final (minutes)" and "unique addresses" have been identified as two of the most crucial indicators of fraudulent transactions.

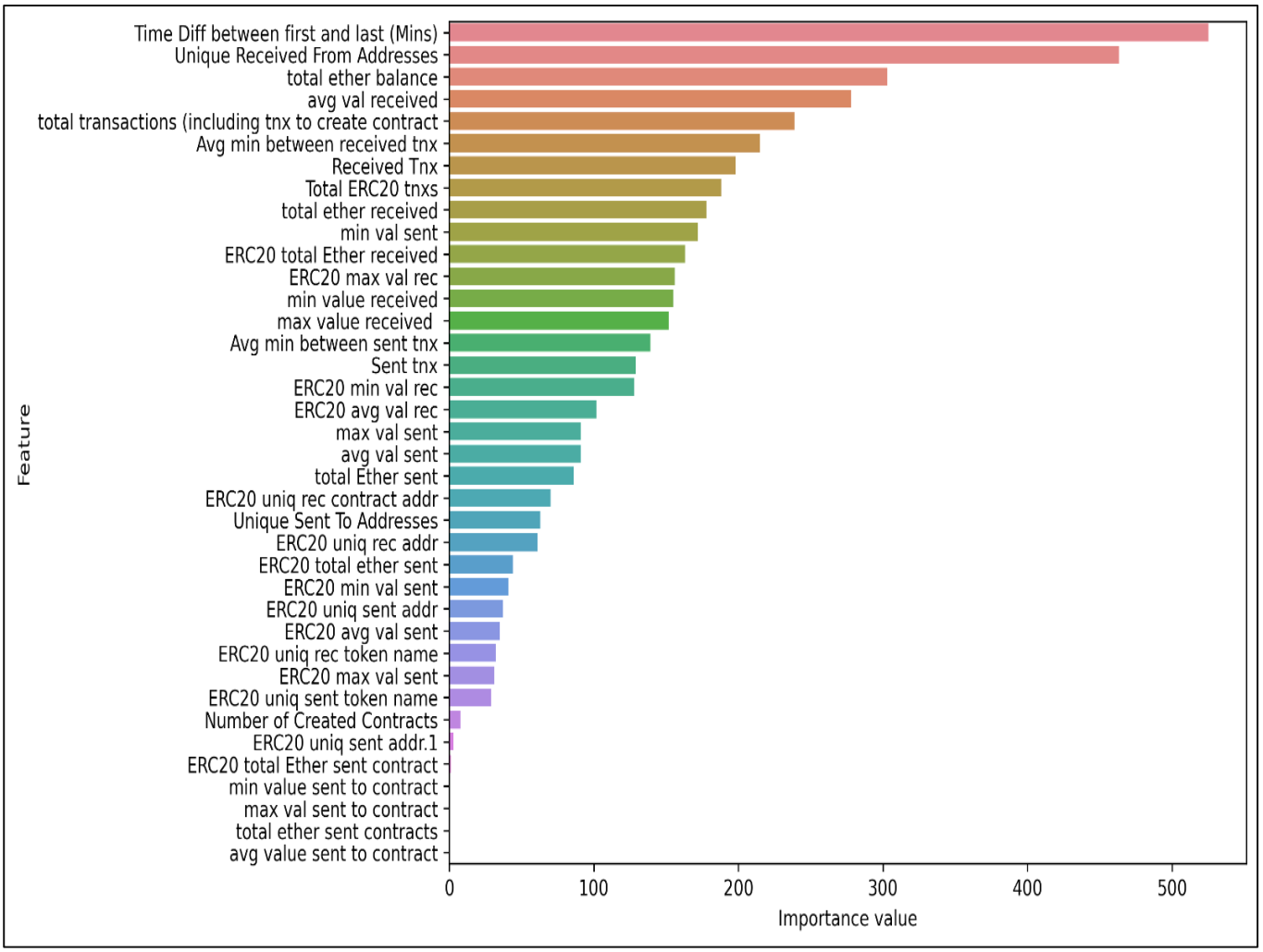
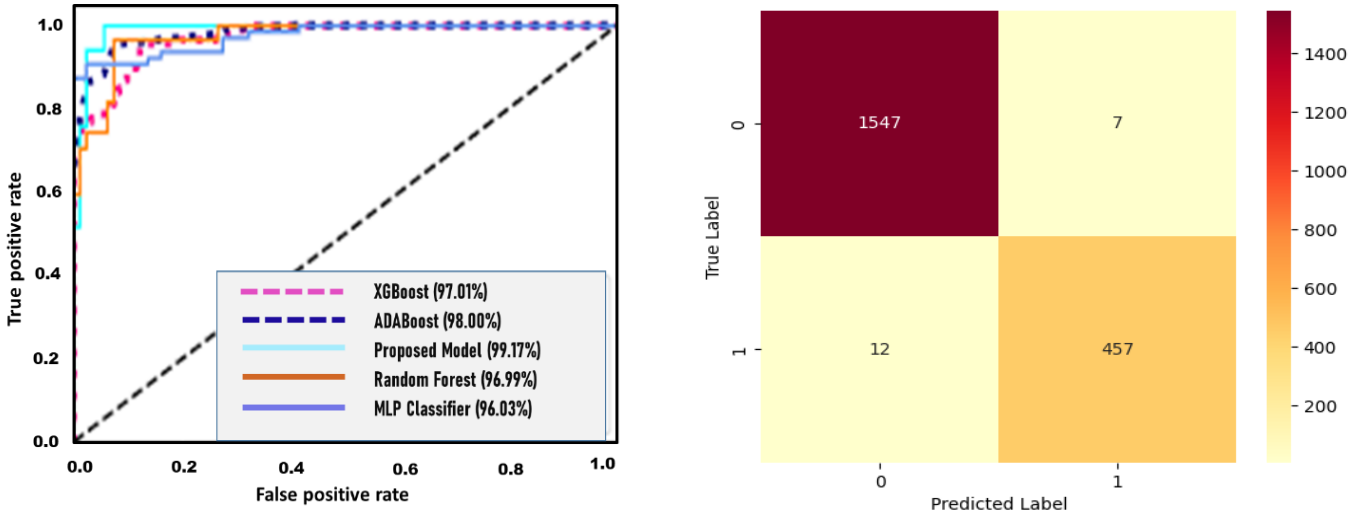


Figure 9. Feature importance comparison.

5.3. Receiver Operating Characteristic of Different Models by Different Classifiers

Training was also utilized to measure accuracy, recall, and precision, in addition to the F1 score. Each model's performance was calculated using the receiver operating characteristic (ROC) curve, which is a graphical depiction of a binary classification model's performance. For varying classification criteria, the ROC curve displays the true positive rate (TPR) vs. the false positive rate (FPR). A threshold is a judgement limit that is used to assess whether a sample is positive or negative. We may trade off the TPR and FPR by modifying the threshold, and the ROC curve demonstrates how the model's performance varies when the threshold is changed. In our case, the proposed model outperforms the ADABoost algorithm (97.01%) on the classification task and identifies the trade-offs between TPR and FPR. Because the model has a higher TPR than the ADABoost algorithm (97.01%) at the same FPR, this would imply that the proposed model (99.17%) was more effective at detecting fraudulent transactions. Figure 10 depicts the comparable findings. According to the ROC of the suggested technique, which has an area under the curve of 98.87%, the classification accuracy of DL employing the given optimization strategy is 98.87%. A confusion matrix, also shown in Figure 10, is used to evaluate the performance of a classification model. It consists of two classes: class 0 represents non-fraudulent transactions and class 1 represents fraudulent transactions. Values in the four cells represent the number of true positive, true negative, false positive, and false negative predictions made by the model, respectively. The true positive value (TP:1547) represents the number of fraudulent transactions that the model correctly identified as fraudulent. The true negative

value (TN:7) represents the number of non-fraudulent transactions that the model correctly identified as non-fraudulent. The false positive value (FP:12) represents the number of non-fraudulent transactions that the model incorrectly identified as fraudulent. The false negative value (FN:457) represents the number of fraudulent transactions that the model incorrectly identified as non-fraudulent. Analyzing the matrix also helps us to gain insight into the performance of the model and identify areas where it may be underperforming. If the false negative value is high, it could indicate that the model is not performing well in detecting fraudulent transactions. On the other hand, if the false positive value is high, it could indicate that the model is generating a large number of false alarms and flagging non-fraudulent transactions as fraudulent.



ROC Curve **Confusion Matrix**

Figure 10. Confusion matrix for proposed algorithm after hyperparameter tuning and ROC.

In comparison to other techniques, the proposed deep learning model performs relatively better (Figure 7). As a result, the proposed method for DL with a hyperparameter configuration provides a strong, deepening framework for the identification of fraudulent Ethereum-based exchanges.

6. Conclusions

This study offered a novel optimization strategy for deep learning classifiers to identify fraudulent Ethereum transactions. An extensive experiment was carried out on fraudulent Ethereum transactions to evaluate the effectiveness of the recommended technique using several machine learning techniques. The main focus of the study was the accuracy of seven different machine learning algorithms, including random forest, logistic regression, the LGBM classifier, and the MLP classifier. There were 43 features and 9841 variables in the dataset that was used to train and test the models. After pre-processing and choosing characteristics, the majority of the algorithms performed satisfactorily. With the maximum accuracy of 99.71% and 98.33%, respectively, the upgraded DL methods and SVC outperformed some of the most sophisticated machine learning systems, according to the data.

However, it was shown that through using specific parameters generated from the hyperparameter tuning of the LGBM model, the accuracy could be raised to the maximum achievable level of 99.17%. The suggested model detects fraudulent transactions by employing the LGBM method. Although this model produced reliable findings, it has one flaw: in order for LGBM to be effective, the dataset must be disproportionately large. The Earthworm Optimizer (EWA), Elephant Herding Optimizer (EHO), Moth Search Algorithm (MSA), Monarch Butterfly Optimization (MBO), and Slime Mold Algorithm (SMA) are

some of the most representative computational intelligence algorithms that can be used to evaluate these issues. One can observe trends in Ethereum transactions, which should be investigated in upcoming research. It is also feasible to build on this research and develop a machine learning model that performs significantly better across all dataset sizes.

Author Contributions: Methodology, R.M.A. and A.D.; Software, K.G.; Validation, K.G.; Formal analysis, R.M., P.K. and A.S.; Investigation, R.M., A.D. and A.S.; Resources, P.K.; Data curation, R.M., K.G., A.D. and A.S.; Writing—original draft, R.M.A. and R.M.; Writing—review & editing, R.M.A. and K.G.; Visualization, P.K. and A.S.; Project administration, P.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Used data set available on internet link is also given in paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Clohessy, L.; Treiblmaier, H.; Acton, T.; Rogers, N. Antecedents of blockchain adoption: An integrative framework. *Strateg. Change* **2020**, *29*, 501–515. [CrossRef]
2. Li, X.; Whinston, A.B. Analyzing cryptocurrencies. *Inf. Syst. Front.* **2020**, *22*, 17–22. [CrossRef]
3. Liu, J.; Serletis, A. Volatility in the cryptocurrency market. *Open Econ. Rev.* **2019**, *30*, 779–811. [CrossRef]
4. Aziz, R.; Baluch, M.; Patel, S.; Ganie, A. LGBM: A machine learning approach for Ethereum fraud detection. *Int. J. Inf. Technol.* **2022**, *14*, 3321–3331. [CrossRef]
5. Leal, F.; Chis, A.E.; Horacio, G.V. Multi-service model for blockchain networks. *Inf. Process. Manag.* **2021**, *58*, 102519–102525. [CrossRef]
6. Aziz, R.M. Cuckoo search-based optimization for cancer classification: A new hybrid approach. *J. Comput. Biol.* **2022**, *29*, 565–584. [CrossRef] [PubMed]
7. Panarello, A.; Tapas, N.; Merlino, G.; Longo, F.; Puliafito, A. Blockchain and IoT integration: A systematic survey. *Sensors* **2018**, *18*, 2575. [CrossRef]
8. Aziz, R.M.; Baluch, M.F.; Patel, S.; Kumar, P. A Machine Learning based Approach to Detect the Ethereum Fraud Transactions with Limited Attributes. *Karbala Int. J. Mod. Sci.* **2022**, *8*, 139–151. [CrossRef]
9. Brauneis, A.; Mestel, R.; Theissen, E. What drives the liquidity of cryptocurrencies? A long-term analysis. *Financ. Res. Lett.* **2021**, *39*, 101537. [CrossRef]
10. Jung, E.; Tilly, M.L.; Gehani, A.; Ge, Y. Data mining-based ethereum fraud detection. In Proceedings of the 2019 IEEE International Conference on Blockchain, Atlanta, GA, USA, 14–17 July 2019; pp. 266–273. [CrossRef]
11. Bartoletti, M.; Carta, S.; Cimoli, T.; Saia, R. Dissecting ponzi schemes on ethereum: Identification, analysis, and impact. *Future Gener. Comput. Syst.* **2020**, *102*, 259–277. [CrossRef]
12. Chen, W.; Zheng, Z.; Ngai, E.C.H.; Zheng, P.; Zhou, Y. Exploiting blockchain data to detect smart ponzi schemes on ethereum. *IEEE Access* **2019**, *7*, 37575–37586. [CrossRef]
13. Vasek, M.; Moore, T. Analyzing the bitcoin ponzi scheme ecosystem. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 101–112. [CrossRef]
14. Ajay, K.; Abhishek, K.; Nerurkar, P.; Ghalib, M.R.; Shankar, A.; Cheng, X. Secure smart contracts for cloud based manufacturing using Ethereum blockchain. *Trans. Emerg. Telecommun.* **2020**, *13*, 4121–4129. [CrossRef]
15. Aljofey, A.; Rasool, A.; Jiang, Q.; Qu, Q. A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain. *Electronics* **2022**, *11*, 2937. [CrossRef]
16. Tan, R.; Tan, Q.; Zhang, P.; Li, Z. Graph neural network for ethereum fraud detection. In Proceedings of the 2021 IEEE International Conference on Big Knowledge (ICBK), Auckland, New Zealand, 7–8 December 2021; pp. 78–85. [CrossRef]
17. Yuan, Q.; Huang, B.; Zhang, J.; Wu, J.; Zhang, H.; Zhang, X. Detecting phishing scams on ethereum based on transaction records. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5. [CrossRef]
18. Ibrahim, R.F.; Elian, A.M.; Ababneh, M. Illicit account detection in the ethereum blockchain using machine learning. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 488–493. [CrossRef]
19. Tsaur, W.J.; Chang, J.C.; Chen, C.L. A highly secure IoT firmware update mechanism using blockchain. *Sensors* **2022**, *22*, 530. [CrossRef]
20. Aziz, R.M.; Sharma, P.; Hussain, A. Machine Learning Algorithms for Crime Prediction under Indian Penal Code. *Ann. Data Sci.* **2022**, 1–32. [CrossRef]
21. Liu, L.; Tsai, W.T.; Bhuiyan, M.Z.; Peng, H.; Liu, M. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Gener. Comput. Syst.* **2022**, *128*, 158–166. [CrossRef]

22. Aziz, R.; Aftab, H.; Prajwal, S.; Kumar, P. Machine learning-based soft computing regression analysis approach for crime data prediction. *Karbala Int. J. Mod. Sci.* **2022**, *8*, 1–19. [CrossRef]
23. Desai, N.P.; Baluch, M.F.; Makrariya, A.; MusheerAziz, R. Image processing model with deep learning approach for fish species classification. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2022**, *13*, 85–99.
24. Chen, L.; Peng, J.; Liu, Y.; Li, J.; Xie, F.; Zheng, Z. Phishing scams detection in ethereum transaction network. *ACM Trans. Internet Technol.* **2020**, *21*, 1–16. [CrossRef]
25. Sreejith, S.; Rahul, S.; Jisha, R. A real time patient monitoring system for heart disease prediction using random forest algorithm. In *Advances in Signal Processing and Intelligent Recognition Systems*; Springer: Cham, Switzerland, 2016; pp. 485–500. [CrossRef]
26. Aziz, R.; Verma, C.; Srivastava, N. A fuzzy based feature selection from independent component subspace for machine learning classification of microarray data. *Genom. Data* **2016**, *8*, 4–15. [CrossRef]
27. Aziz, R.M. Nature-inspired metaheuristics model for gene selection and classification of biomedical microarray data. *Med. Biol. Eng. Comput.* **2022**, *60*, 1627–1646. [CrossRef] [PubMed]
28. Aziz, R.M.; Desai, N.P.; Baluch, M.F. Computer vision model with novel cuckoo search based deep learning approach for classification of fish image. *Multimed. Tools Appl.* **2022**, 1–20. [CrossRef]
29. Farrugia, S.; Ellul, J.; Azzopardi, G. Detection of illicit accounts over the ethereum blockchain. *Expert Syst. Appl.* **2020**, *150*, 113318. [CrossRef]