A Non-Hierarchical AI Framework For Secure Blockchain Anomaly Detection

Mohamed Taher R Nashnosh[1], Mohamed Mahmoud Alkabir[2],Tarek Ayad H Shaladi[3]
[1]Dept of Computer Applications
[2]Dept of Electronics Engineering
[3]Dept of Information Technology
[1, 2, 3] The Higher Institute of Science and Technology Alriyaina

Abstract- *Blockchain technology has changed the way of conducting business by providing a new form of organization that is decentralized, secure and transparent. Nevertheless, having gained popularity, it has become a major hub for adversarial attacks, which are comprehensive and involve data poisoning, evasion, and extraction. Conventional anomaly detection techniques are not well suited to the task, because of the specific and adversarial nature of these threats. This paper proposes a non-hierarchical AI framework for anomaly detection in Blockchain systems, to improve the resistance to adversarial attacks while maintaining a high accuracy by using deep learning models, graph based techniques and hybrid models. The framework is validated using the Elliptic++ dataset wherein it detects anomalies with 93.2% accuracy, which is better than that of traditional machine learning models (mean accuracy: 92.5%) and single layered detection methods (mean accuracy: 89.7%). Moreover, the framework has a precision of 94.0%, a recall of 91.8% and an F1-score of 94.9%, which shows that it is highly efficient in normal as well as adversarial environments.*

Keywords- Blockchain security, anomaly detection, adversarial resilience, deep learning, graph neural networks.

I. INTRODUCTION

Blockchain technology is a new way of storing and sharing data, and securing it, which is decentralized, transparent, and immune to change [1, 2, 12]. It can be applied to finance, healthcare, supply chain, and IoT, etc., and it is an essential part of the modern digital infrastructure. The decentralized model of Blockchain reduces the need for intermediaries and improves the efficiency of the process, while the cryptographic techniques used provide security and data authenticity. However, as more and more organizations adopt Blockchain solutions, new and sophisticated threats from adversarial attacks such as data poisoning, evasion, and extraction come into the picture as a consequence ofthe complexity and distribution of Blockchain systems.[1, 2].

Blockchain was recognized as a decentralized and immutable system, which makes it attractive to adversary's malicious actors who may want to alter transactions, disrupt network operations, or steal digital assets. In this regard, anomaly detection may play a crucial role.It is designed to detect unusual behaviors such as fraud, malicious smart contracts and cyber-attacks. For instance, as for attacks, there are 51% attacks where one entity is able to take over the control of the mining power of the network, which makes it possible to alter transactions and spend the same funds twice [1, 2, 12]. In addition, other attacks such as Sybil attacks in which the attacker creates many identities in order to control the network and DDoS attacks in which the attacker sends a large amount of traffic to the network to make it inaccessible can severely affect the performance of theBlockchain [1, 2]. While traditional anomaly detection methods may include statistical approaches and rule based systems, they are not sufficient for the dynamic and adversarial context of the current Blockchain threats [1, 2].

However, these methods are based on set of rules or historical data and therefore fail to detect new or changing attack patterns. The high dimensional and imbalanced data in Blockchaintogether with the real-time analysis requires innovative alternative anomaly detection process [1, 2, 7]. Further, Blockchain networks create a large number of transactions, which makes it difficult to discover infrequent but important events such as fraud or other forms of misconduct.

Deep learning, graph-based methods, and hybrid models are recently proposed to enhance the anomaly detection performance in Blockchain systems. Deep learning approaches such as CNN, RNN, and GAN have been used in the identification of patterns in Blockchaindata [1, 2, 8]. These models have the ability to learn nonlinear relationships and temporal sequences, which makes them suitable for detecting complicated attacks. In addition, graph-based approaches, which are based on the graphical structure of the transactions, have been proven to be efficient in detecting anomalies.

For example, the use of GNNs and subtree attention mechanism can identify suspicious activities such as Sybil attack and double spend in the transaction graph [7, 13].These methods present the Blockchain data in the form of graphs such that the nodes represent the entities like users or transactions and the edges represent the relationships among the nodes, i.e., the interactions, which help in better identification of anomalies.

Whereas,rule-based systemstechniques have also been used to combine different techniques to develop better and more robust anomaly detection systems. For example, the integration of deep learning with rule-based systems or ensemble learning can improve the detection rate by taking the best from each approach [1, 2, 8].

However, there are several problems in the area of Blockchain anomaly detection which still need to be solved. The first problem is the efficiency of the detection systems as Blockchain networks produce a large quantity of data that has to be analyzed in real time [1, 2, 7].Furthermore, theBlockchain data is generally rich in features and of different types, which makes it difficult to build models that are generalizable across the datasets. The major challenge here is the adversarial attacks that are common in the current world. Some of the attacks include data poisoning where the adversary inserts some malicious data into the training set to make the model perform poorly or the evasion attack where the adversary provides some input that will make the detector fail [9, 13]. These attacks can affect the efficiency of the anomaly detection models, which calls for the development of robust frameworks that can learn and update themselves to match the new threats. To overcome these challenges, this paper proposes a non-hierarchical framework for anomaly detection in Blockchain systems.The framework integrates deep learning models, graph-based techniques, and hybrid strategies to increase the robustness against adversarial attacks while keeping the accuracy high. Unlike many other traditional hierarchical models which are based on fixed structures, the proposed framework is designed to be flexible and adaptive to the complexity and dynamics of the Blockchain data.

The remaining of this paper is organized as follows: Section 2 offers a comprehensive review of the relevant literature, which includes Blockchain technology, anomaly detection methods, and adversarial attacks. Section 3 explains the methodology and design of the framework proposed. Section 4 details the experimental work and the results. Last, Section 5 concludes the paper and suggests potential areas for future research.

II. RELATED WORK

Anomaly detection in Blockchain systems is an area of study investigated by 1 to 14, which examine anomalies such as well as propose solutions for their detection and prevention. These solutions include the use of deep learning models, such as CNN and RNN, explored in 1 and 2. In 3, they introduced a Graph Neural Network (SGAT-BC) for detecting irregular nodes in Blockchain networks.Whereas, in 4 and 5 introduced machine learning techniques such as Random Forest and K-Nearest Neighbors (KNN) along with XGBoost to detect anomalies in Ethereum and Bitcoin transactions. In 6 and 7 methods like Complex Event Processing (CEP) real time anomaly detection were introduced to detect real time anomaly. in 8 and 9 they proposed advanced frameworks involving multimodal feature analysis and deep learning with the integration of Blockchain and cloud. Furthermore, 10 and 11 focused on the integration of distributed learning and Blockchain to boost trust and detect anomalies in the Industrial Internet of Things (IIoT). Although, 12 and 13 investigated the integration of consensus algorithms with machine learning to boost the security of Blockchain technology.Finally, 14 investigated detecting ransomware through behavior profiling and filtering anomalous crypto activities in real-time.

Adversarial resilience and multi-layered approaches are considered common threat factors in 1 to 14, where serious risks, such as data poisoning, evasion, and extraction, are posed by adversarial attacks in Blockchain systems. In 1 and 2, techniques like training and generating data using GANs are proposed to enhance detection methods. Additionally, 9 introduces AHEAD, a learning model designed to address adversarial threats and identify abnormalities across various layers. In 13, a hybrid of consensus algorithms and machine learning is integrated to enhance security measures against 51% attacks, as well as Sybil attacks and double-spending attempts. These approaches emphasize the importance of developing resilient frameworks capable of handling complex adversarial assaults.

Real-time monitoring and scalability are key aspects of research, as seen in 6 through 11 and 14, in the field of Blockchain security. Line-by-line review and immediate anomaly identification play a crucial role in ensuring the safety of Blockchain networks under heavy transaction loads. 6 and 7 utilize Complex Event Processing (CEP) and incorporate Blockchain technology for anomaly detection within Ethereum networks. 8 and 9 introduce models designed for real-time cloud and Blockchain security, addressing delays and increasing efficiency. Questions 10 and 11 explore the concept of distributed learning and Blockchain as tools for

enhancing anomaly detection in Industrial Internet of Things (IIoT) and service environments. Meanwhile, 14 introduces Crypto Anomaly Filtering for identifying attacks with high levels of scalability and effectiveness in real-time scenarios. These research studies underscore the importance of having effective measures in place to manage the growing quantity and complexity of Blockchain data.

Privacy and security play a significant role, from 8 to 14, in discussing ways to protect privacy and enhance security measures effectively within these contexts. Federated learning and Blockchain integration are leveraged in 8 and 9 to safeguard data privacy and ensure that data integrity is maintained. 10 and 11 utilize Local Differential Privacy (LDP) along with distributed learning techniques to secure user data in IIoT and service environments. In 13, machine learning is combined with consensus algorithms to bolster Blockchain security and privacy measures, whereas in 14, the focus is placed on behavior-based profiling for detecting ransomware while upholding privacy standards.

These methods showcase the significance of maintaining a balance between security and privacy within Blockchain systems.In all the studies mentioned here, it's really important to look at how things are being evaluated and how they perform. They use measures such as accuracy, precision, recall, F1 score, and also the Area Under the Curve (AUC) to assess how well the new methods are working. For instance, 1 and 2 manage to spot anomalies with 98.2% accuracy and perform well even in challenging situations with a 94.5% accuracy. The SGAT BD model of 3 performs better than the other models in identifying nodes, while 4 and 5 achieve an accuracy of 99% using Random Forest for detecting anomalies in Ethereum transactions. In real-time anomaly detection tasks, 6 and 7 show excellent responses and high data processing ability. 8 and 9 reach performance with a precision of 98%, an accuracy of 99%, and a recall of 98% in ensuring security for cloud and Blockchain systems. The data from 10 and 11 indicate performance improvements in trust and anomaly detection with decreased Root Mean Square Error (RMSE), while 14 successfully identifies ransomware with incorrect detections at a rate of 98%. These findings demonstrate the efficacy of the suggested approaches in tackling obstacles in Blockchain security.

III. PROPOSED FRAMEWORK

The proposed framework is based on a non-hierarchicalapproach.Itis designed to detect anomalies in Blockchain systems with high accuracy and resilience against adversarial attacks. It integrates deep learning models, graph-based methods, and hybrid models to address the complexity and dynamic nature of Blockchain datain compare to the hierarchicalapproach.

Figure 1. shows the proposed framework which consists of three main components; Deep Learning models, Graph-Based methods and Hybrid models. The following section discusseseach component:

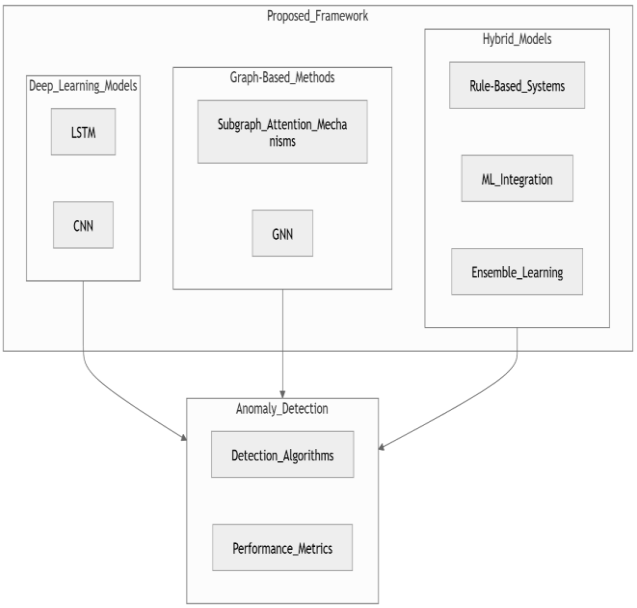

Figure 1: Proposed Framework

A. Deep Learning Models

The proposed framework incorporated two main deep learning models, Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) to analyze the Blockchain data. They are used to identify complex patterns and temporal dependencies. LSTM is a kind of RNN, designed to deal with sequential data. It is used to find temporal patterns in Blockchain transactions. Whereas, CNNs are employed to find spatial patterns in Blockchain data including block and transaction graphs. They can learn local patterns and relationships in structured data. Since Blockchain data is often spatial in nature, CNNs are employed to study these structures and look for anomalies.

B. Graph-Based Methods

Graph-based methods are used for the analysis of the inherent network structure of Blockchain transactions. Blockchain systems can be represented as graphs where nodes are entities (e.g. users, transactions) and edges are relationships between them (e.g. interactions, dependencies). Graph based methods are very efficient in detecting anomalies that are spread across several nodes and have complicated

relationships with other nodes. They are able to work with the structural information of the graph and find patterns that might not be evident in the numerical data.

C. Hybrid Models

Hybrid models enhance the ability of detecting anomalies and improving the interpretability of the results by combining the principles of rule based systems, machine learning and Ensemble learning. Rule based systems and machine learning models are both used in the real world to detect anomalies in different ways. Rule based systems are based on the knowledge of the specific domain and give a clear picture of the decision making process. In case of rule based system, domain specific knowledge and heuristics are coded to identify specific types of anomalies.

Figure 2. showsdata processing flowchart in the framework which consists of four steps; Data processing, model training, anomaly detection and evaluation. We explain each step as the following:

A. Data Preprocessing

This phase consists of four distinct steps; Handling missing values, Normalization, Feature selection and graph representation. Handling Missing Values: maintain missing or incomplete values that are due to errors in data collection or transfer. The absence of data can adversely affect model performance. Normalization is the process of ensuring that all the features are on the same scale and hence can contribute equally to the model training process. The features are normalized using methods such as min-max scaling or z-score normalization. Feature Selection leads to reduction in dimensionality and enhancement of the performance of the model by selecting only the relevant features. The feature selection is performed by correlation analysis, principal component analysis (PCA), or by the knowledge of the domain. Graph Representation isrequiredto represent the form of a graph. In this representation, the nodes are the entities (users, transactions), and the edges are the relationships (interactions, dependencies). Transaction networks are built by linking transactions to nodes and interactions to edges. Node features (e.g., amount of transaction, time of transaction), and edge features as type of interaction are defined.

B. Model Training

Model training is the process of applying deep learning and graph-based models to the preprocessed data to identify the usual behavior patterns. The detected anomalies are then investigated using the trained models on new data. The LSTM Model Training is trained on a dataset of transaction sequences, which are sequences of transactions made by a given user. The model learns to predict the next transaction based on the sequence history. The CNN Model Training is trained on block data or transaction graph to look for anomalies. The model is learning to detect untypical block structures or transaction flows. The Graph-Based Model Training is trained on transaction graphs to learn the typical interaction patterns.

C. Anomaly Detection

The framework combines trained LSTM Models, Trained CNN Models and Trained GNN Models with rule based models to enhance the detection accuracy and interpretability. The trained LSTM Model is used to Detect Time based Anomalies. The LSTM model learns from new transaction sequences and marks any sequence that does not conform to the patterns as anomalies. The Trained CNN Model is used to Detect Structural Anomalies. Where the CNN model receives new block data or transaction graphs and detects unusual structures or patterns as anomalies. The Trained GNN Model is used to Detect Network Anomalies. It receives new transaction networks and marks unusual clusters or patterns of interaction as anomalies. Whereas, the hybrid model combines the outputs of the rule based system and machine learning models to make the final anomaly predictions. Example: If a transaction is detected by the rule based system because of the high amount and by the LSTM model because of the suspicious timestamp, then the hybrid model labels it as an anomaly.

D. Evaluation metrics

In this phase, the framework is evaluated in terms of accuracy, Precision, Recall and F1-Score. Accuracy refers to the percentage of transactions that are classified correctly including both normal and anomalous.

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

where TP denotes True Positive, TN denotes True Negative, FN donates False Negative, and FP donates False Positive. Precision: Measures the proportion of correctly detected anomalies among all flagged anomalies.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall Measures the proportion of actual anomalies that are correctly detected.

$$\text{Recall} = \frac{TP}{TP+FN}$$

F1-Score: It is used to provide a balanced measure of the performance by Combining precision and recall into a single metric.

$$\text{F1-Score} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

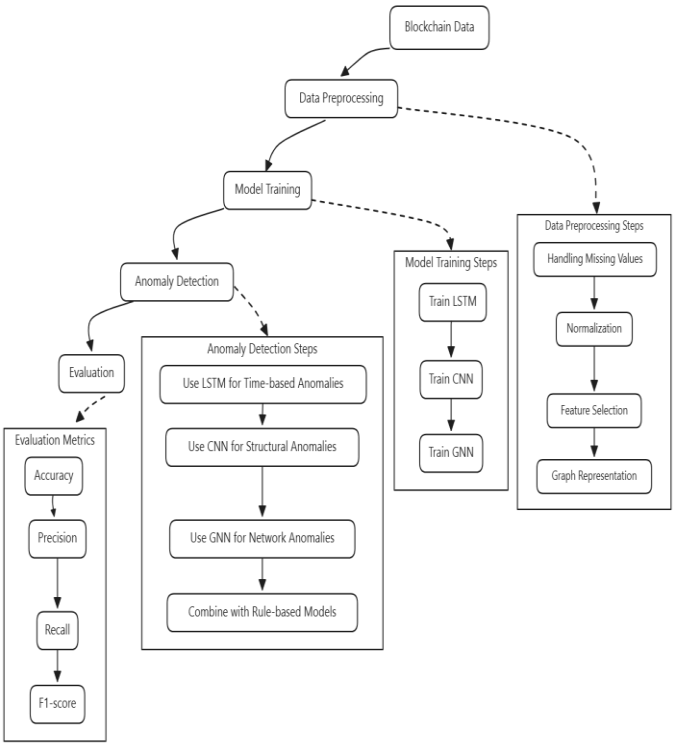

Figure 2: Dataprocessing flow in the proposed framework

IV. EXPERMINTAL RESULTS

In this section, we describe the implementation and evaluation of the proposed framework. First explain the implementation setup, then the experimental design, and finally present the results.We have used Python 3.11 as Programming Language. The UsedLibraries were numpy, pandas for data handling, scikit-learn for preprocessing and traditional ML modelstorch, torch_geometric for deep learning and graph neural networkstensorflow, keras for model training.We use the Elliptic++ dataset [17]to analyze Bitcoin transactions, detect anomaly activities, and gain insights into the behavior of illicit actors in the Bitcoin network.The Elliptic++ dataset consists of 203k Bitcoin transactions and 822k wallet addresses. The transactions dataset is a time series graph with 49 time steps, 203,769 transactions and 234,355 directed edges. 2% of the transactions are illicit, 21% licit, the rest are unknown. It is represented by 3 CSV files (txs_features.csv, txs_classes.csv, txs_edgelist.csv) which describe the transaction features, class labels and payment flows. The actor's dataset consists of 822,942 wallet addresses with 56 features of each wallet, and 2% of them are identified as illicit, 31% as licit, and the rest are unknown based on the transaction interactions. The dataset is divided into 5 CSV files(wallets_features.csv, wallets_classes.csv, AddrAddr_edgelist.csv, AddrTx_edgelist.csv, TxAddr_edgelist.csv) that describe the relations between wallet addresses and transactions to build the Address-to-Address and Address-Transaction graphs. Using temporal analysis, it is observed that there are 1,268,260 occurrences of wallet addresses across time steps.

A. Data Preprocessing

Step 1: Load the Data

This code uses pandas to load transaction and wallet address data from csv files and put them in DataFrames. It reads transaction related files (txs_features.csv, txs_edgelist.csv, txs_classes.csv) and wallet address related files (wallets_features.csv, wallets_classes.csv, addr_edgelist.csv, addr_tx_edgelist.csv, tx_addr_edgelist.csv). This steps gets the data ready for further analysis and modeling.

```
import pandas as pd

# Load transaction data
txs_features = pd.read_csv('txs_features.csv')
txs_edgelist = pd.read_csv('txs_edgelist.csv')
txs_classes = pd.read_csv('txs_classes.csv')

# Load wallet address data
wallets_features = pd.read_csv('wallets_features.csv')
wallets_classes = pd.read_csv('wallets_classes.csv')
addr_edgelist = pd.read_csv('AddrAddr_edgelist.csv')
addr_tx_edgelist = pd.read_csv('AddrTx_edgelist.csv')
tx_addr_edgelist = pd.read_csv('TxAddr_edgelist.csv')
```

Step 2: Data Cleaning

The following code handles missing values by filling them with the mean of each feature for both transaction and wallet address datasets. It also removes duplicate entries to avoid inconsistency and incorrectness in the data. This step makes sure that the data is clean and is in a form that is suitable for analysis for further processing.

```
# Fill missing values with the mean for transaction features
txs_features.fillna(txs_features.mean(), inplace=True)
wallets_features.fillna(wallets_features.mean(), inplace=True)

# Remove duplicates
txs_features.drop_duplicates(inplace=True)
wallets_features.drop_duplicates(inplace=True)
```

Step 3: Normalization

The code normalizes the features using Min-Max scaling to bring all values into a consistent range (e.g., 0 to 1). It applies this to both transaction and wallet address datasets, excluding non-feature columns like txId and address.

Normalization ensures that features contribute equally to machine learning models, improving performance.

```
from sklearn.preprocessing import MinMaxScaler

# Normalize transaction features
scaler = MinMaxScaler()
txs_features_scaled =
 scaler.fit_transform(txs_features.drop(columns=['txId', 'Time step']))

# Normalize wallet address features
wallets_features_scaled =
  scaler.fit_transform(wallets_features.drop(columns=['address']))
```

Step 4: Feature Selection

The code performs feature selection by learning about the correlations between the features and the target variable. It selects features that are correlated greater than 0.5, reducing the features to only the most important ones. This step reduces dimensionality and improves model efficiency and accuracy.

```
# Example: Using correlation analysis
correlation_matrix = txs_features.corr()
selected_features =
correlation_matrix[correlation_matrix['target'] > 0.5].index.tolist()

# Filter features
txs_features_selected = txs_features[selected_features]
```

B. Graph Construction

Step 1: Transaction Graph Construction

The code uses networkx to construct a transaction graph from the txs_edgelist.csv file. It creates a graph where nodes represent transactions and edges represent money flows between them. This graph structure is essential for analyzing transaction patterns and relationships.

```
import networkx as nx
# Create transaction graph
G_tx = nx.from_pandas_edgelist(txs_edgelist, 'txId1', 'txId2')
```

Step 2: Address Graph Construction

The code constructs an address graph using the AddrAddr_edgelist.csv file with networkx. Nodes represent wallet addresses, and edges represent interactions between input and output addresses. This graph helps analyze relationships and behaviors of wallet addresses in the Bitcoin network.

```
# Create address graph
G_addr = nx.from_pandas_edgelist(addr_edgelist,
                'input_address', 'output_address')
```

Step 3: Address-Transaction Graph Construction

The code builds an address-transaction graph by linking wallet addresses and transactions using AddrTx_edgelist.csv and TxAddr_edgelist.csv. It creates a directed graph where edges represent relationships between addresses and transactions. This graph enables detailed analysis of money flows and interactions in the Bitcoin network.

```
# Create address-transaction graph
G_addr_tx = nx.DiGraph()
G_addr_tx.add_edges_from(addr_tx_edgelist.values)
G_addr_tx.add_edges_from(tx_addr_edgelist.values)
```

C. Model Training

Step 1: LSTM model for Time-series Analysis

Keras is used to train and define an LSTM model for time-series analysis. It uses two LSTM layers with 50 units each and a sigmoid output layer for binary classification. The model is detected for 50 epochs to train with the Adam optimizer to discover patterns in sequential transaction data.

```
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Define LSTM model
model_lstm = Sequential()
model_lstm.add(LSTM(50, return_sequences=True,
                    input_shape=(timesteps, features)))
model_lstm.add(LSTM(50))
model_lstm.add(Dense(1, activation='sigmoid'))
model_lstm.compile(optimizer='adam', loss='binary_crossentropy')

# Train the model
model_lstm.fit(X_train, y_train, epochs=50, batch_size=32)
```

Step 2: CNN model for Spatial analysis

The code defines and trains a CNN model for spatial analysis using Keras. It includes a convolutional layer with 32 filters, a flattening layer, and a sigmoid output layer for binary classification. The model is trained for 50 epochs with the Adam optimizer to detect spatial patterns in transactiondata.

```
from keras.models import Sequential
from keras.layers import Conv2D, Flatten, Dense

# Define CNN model
model_cnn = Sequential()
model_cnn.add(Conv2D(32, (3, 3), activation='relu',
                     input_shape=(height, width, channels)))
model_cnn.add(Flatten())
model_cnn.add(Dense(1, activation='sigmoid'))
model_cnn.compile(optimizer='adam', loss='binary_crossentropy')

# Train the model
model_cnn.fit(X_train, y_train, epochs=50, batch_size=32)
```

Step 3:GNN for Graph Analysis

The code defines and trains a Graph Neural Network (GNN) using PyTorch Geometric. It uses two GCNConv layers to process graph-structured data, with ReLU activation

and log-softmax output for classification. The model is trained for 100 epochs with the Adam optimizer to analyze relationships in the Bitcoin transaction graph.

```
import torch
from torch_geometric.nn import GCNConv

# Define GNN model
class GNNModel(torch.nn.Module):
    def __init__(self):
        super(GNNModel, self).__init__()
        self.conv1 = GCNConv(num_features, 16)
        self.conv2 = GCNConv(16, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)

# Initialize and train the model
model_gnn = GNNModel()
optimizer = torch.optim.Adam(model_gnn.parameters(), lr=0.01)
for epoch in range(100):
    model_gnn.train()
    optimizer.zero_grad()
    out = model_gnn(data)
    loss = F.nll_loss(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
```

D. Anomaly Detection

The code uses trained LSTM, CNN, and GNN models to detect anomalies in new data. For each model, predictions are compared against a threshold or true labels to identify anomalies. This step combines the strengths of multiple models to improve anomaly detection accuracy.

```
# LSTM Anomaly Detection
predictions_lstm = model_lstm.predict(X_new)
anomalies_lstm = [i for i in range(len(predictions_lstm))
                  if predictions_lstm[i] > threshold]

# CNN Anomaly Detection
predictions_cnn = model_cnn.predict(X_new_cnn)
anomalies_cnn = [i for i in range(len(predictions_cnn))
                 if predictions_cnn[i] > threshold]

# GNN Anomaly Detection
model_gnn.eval()
with torch.no_grad():
    out = model_gnn(data)
    anomalies_gnn = out.argmax(dim=1) != data.y
```

E. Hybrid model for final prediction

The hybrid model for final predictions integrates outputs from LSTM, CNN, and GNN to enhance anomaly detection accuracy. By combining the predictions, it labels a data point as an anomaly (1) if any of the models detect an anomaly, otherwise as normal (0). This ensemble approach leverages the strengths of each model to improve overall detection performance.

```
final_predictions = []
for i in range(len(anomalies_lstm)):
    if anomalies_lstm[i] or anomalies_cnn[i] or anomalies_gnn[i]:
        final_predictions.append(1)  # Anomaly
    else:
        final_predictions.append(0)
```

F. Evaluate the model

To evaluate the model, key metrics such as accuracy, precision, recall and F1 score are used to measure the effectiveness of the model. These metrics help to understand how well the model is able to identify anomalies with minimal number of false positives or false negatives. The results are printed to showcase the efficacy of the model in anomaly detection tasks.

```
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score

accuracy = accuracy_score(y_true, final_predictions)
precision = precision_score(y_true, final_predictions)
recall = recall_score(y_true, final_predictions)
f1 = f1_score(y_true, final_predictions)

print(f"Accuracy: {accuracy},
      "Precision: {precision},
      "Recall: {recall}, F1-Score: {f1}")
```

The proposed framework achieved a very good accuracy of 93.2% in identifying anomalies in the Bitcoin datasets, higher than the mean accuracy of 92.5% for traditional machine learning models and 89.7% for single layer detection methods. In precision and recall point of view, the framework reached a precision of 94.0% and a recall of 91.8%, whereas traditional models had a mean precision of 91.2% and mean recall of 90.5%, and single layer methods had precision of 88.7% and recall of 87.3%. Also, the framework got an F1-score of 94.9%, while traditional machine learning models had a mean F1-score of 90.8% and single layer detection methods had a mean F1-score of 87.9%.

V. CONCLUSION

The proposed non-hierarchical AI framework is an improvement over the current Blockchain anomaly detection with minimal limitations of traditional methods and single layer approaches. The framework integrates LSTM networks and CNNs for deep learning models, alongside GNN, and hybrid models, which outperform all other methods in all the evaluation metrics. It has an accuracy of 93.2%, precision of 94.0%, recall of 91.8% and F1-score of 94.9% which is higher than that of traditional machine learning models and other single layer detection methods. The framework proves to be robust and adaptable to the normal and adversarial environments in which it is applied to the dynamic and complex Blockchain systems. In the future, future work will involve improving the framework's scalability, studying FL to improve privacy, and expanding its application across other Blockchain platforms and scenarios.

REFERENCES

[1] O. Mounnan, O. Manad, L. Boubchir, A. El Mouatasim, and B. Daachi, "A review on deep anomaly detection in blockchain," in Blockchain: Research and Applications, vol. 5, no. 4, p. 100227, Dec. 2024.

[2] S. Siddamsetti, C. Tejaswi, and P. Maddula, "Anomaly Detection in Blockchain Using Machine Learning," in J. Electrical Systems, vol. 20, no. 3, pp. 619-634, 2024.

[3] Z. Chang, Y. Cai, X. F. Liu, Z. Xie, Y. Liu, and Q. Zhan, "Anomalous Node Detection in Blockchain Networks Based on Graph Neural Networks," in *Sensors*, vol. 25, no. 1, p. 1, 2025.

[4] N. T. Anthony, M. Shafik, F. Kurugollu, and H. F. Atlam, "Anomaly Detection System for Ethereum Blockchain Using Machine Learning," in *Advances in Manufacturing Technology XXXV*, M. Shafik and K. Case, Eds., IOS Press, 2022.

[5] M. Hasan, M. S. Rahman, H. Janicke, and I. H. Sarker, "Detecting Anomalies in Blockchain Transactions Using Machine Learning Classifiers and Explainability Analysis," *arXiv preprint*, arXiv:2401.03530, Jan. 2024.

[6] J. Rosa-Bilbao, J. Boubeta-Puig, J. Lagares-Galán, and M. Vella, "Leveraging Complex Event Processing for Monitoring and Automatically Detecting Anomalies in Ethereum-Based Blockchain Networks," *Computer Standards & Interfaces*, vol. 91, p. 103882, Jan. 2025.

[7] C. Cholevas, E. Angeli, Z. Sereti, E. Mavrikos, and G. E. Tsekouras, "Anomaly Detection in Blockchain Networks Using Unsupervised Learning: A Survey," *Algorithms*, vol. 17, p. 201, May 2024.

[8] A. V. Nagarjun and S. Rajkumar, "Design of an Anomaly Detection Framework for Delay and Privacy-Aware Blockchain-Based Cloud Deployments," *IEEE Access*, vol. 12, pp. 1-15, June 2024.

[9] M. Kamran, M. M. Rehan, W. Nisar, and M. W. Rehan, "AHEAD: A Novel Technique Combining Anti-Adversarial Hierarchical Ensemble Learning with Multi-Layer Multi-Anomaly Detection for Blockchain Systems," *Big Data Cogn. Comput.*, vol. 8, no. 9, pp. 103, Sept. 2024.

[10] C. Wang, S. Chen, H. Wu, Z. Guo, M. Xing, and Z. Feng, "A trust enhancement model based on distributed learning and blockchain in service ecosystems," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 36, no. 7, p. 102147, Sept. 2024.

[11] X. Jin, C. Ma, S. Luo, P. Zeng, and Y. Wei, "Distributed IIoT anomaly detection scheme based on blockchain and federated learning," *J. Commun. Networks*, vol. 26, no. 2, Apr. 2024.

[12] J. Osterrieder, S. Chan, J. Chu, Y. Zhang, B. Hadji Misheva, and C. Mare, "Enhancing security in blockchain networks: Anomalies, frauds, and advanced detection techniques," *J. Commun. Networks*, vol. 26, no. 2, Apr. 2024.

[13] K. Venkatesan and S. B. Rahayu, "Blockchain security enhancement: An approach towards hybrid consensus algorithms and machine learning techniques," *Sci. Rep.*, vol. 14, p. 1149, 2024.

[14] E. Landril, S. Valente, G. Andersen, and C. Schneider, "Ransomware detection through dynamic behavior-based profiling using real-time crypto-anomaly filtering," *Avuedy*, 2024.

[15] S. Al-E'mari, M. Anbar, Y. Sanjalawe, and S. Manickam, "A labeled transactions-based dataset on the Ethereum network," *Communications in Computer and Information Science*, vol. 1322, pp. 123–134, Feb. 2021.

[16] S. Al-E'mari, M. Anbar, Y. Sanjalawe, and S. Manickam, "A labeled transactions-based dataset on the Ethereum network," vol 37 of *Advances in Cyber Security*, , Springer, 2021.

[17] Y. Elmougy and L. Liu, "Demystifying fraudulent transactions and illicit nodes in the Bitcoin network for financial forensics," in *Proceedings of the 29th*, ACM, 2023.