Dynamic Feature Fusion: Combining Global Graph Structures and Local Semantics for Blockchain Phishing Detection

Zhang Sheng*, Liangliang Song*, Yanbin Wang†

***Abstract*—The advent of blockchain technology has facilitated the widespread adoption of smart contracts in the financial sector. However, current phishing detection methodologies exhibit limitations in capturing both global structural patterns within transaction networks and local semantic relationships embedded in transaction data. Most existing models focus on either structural information or semantic features individually, leading to suboptimal performance in detecting complex phishing patterns. In this paper, we propose a dynamic feature fusion model that combines graph-based representation learning and semantic feature extraction for blockchain phishing detection. Specifically, we construct global graph representations to model account relationships and extract local contextual features from transaction data. A dynamic multimodal fusion mechanism is introduced to adaptively integrate these features, enabling the model to capture both structural and semantic phishing patterns effectively. We further develop a comprehensive data processing pipeline, including graph construction, temporal feature enhancement, and text preprocessing. Experimental results on large-scale real-world blockchain datasets demonstrate that our method outperforms existing benchmarks across accuracy, F1 score, and recall metrics. This work highlights the importance of integrating structural relationships and semantic similarities for robust phishing detection and offers a scalable solution for securing blockchain systems. Our code is available at https://github.com/dcszhang/Dynamic_Feature**

***Index Terms*—Blockchain, Fraud Detection, Multimodal Fusion, Security**

I. Introduction

BLOCKCHAIN technology has developed rapidly in recent years and has triggered far-reaching changes in several fields, especially in the financial industry [1]. However, as the popularity of blockchain applications grows, so does the significant increase in fraudulent behaviors it has brought about, with serious implications for society [2]. Blockchain technology, due to its decentralization and transparency, has become a tool for unscrupulous individuals to exploit, although it provides greater security and efficiency in financial transactions [3]. For example, the application of blockchain technology in the supply chain is seen as an effective means to enhance transparency and traceability, but it also faces a crisis of social trust due to fraudulent behavior [4]. In addition, the increase in fraudulent and illegal activities poses new challenges to the global economy as blockchain expands and its applications grow, especially in high-risk financial transactions [5]. Therefore, despite its enormous potential, blockchain technology comes with social and regulatory issues that need to be addressed to ensure its safe and sustainable development [6].

Phishing detection in Ethereum has long relied on Graph Neural Networks (GNNs) [7] to model fund flows in transaction graphs. While these methods [8] leverage graph structure learning to capture transaction topologies, the binary nature of transaction relationships and GNN's neighbor sampling strategy (presence/absence) fail to extract individual account-level behavioral patterns, such as periodic transfers, fixed counterparty preferences, or transaction bursts in specific time windows. Recent approaches using contextual modeling of account transaction sequences via sequence models (e.g., Transformer, LSTM) capture transaction context from full records, addressing the limitations of graph-based methods but missing topological information. In addition, the current research lacks insights into two key aspects:

1) **Local semantic similarity information:** In blockchain transaction data, legitimate and phishing accounts show distinct local patterns. Normal accounts exhibit random behavior—infrequent transactions with irregular amounts and intervals, resulting in weak semantic correlations. Phishing accounts, however, display consistent patterns—frequent transactions in short bursts, with similar amounts or repetitive actions, driven by automated fraud to move funds or obscure trails. Current detection methods struggle to capture these local semantic similarities, limiting their accuracy in identifying fraud.
2) **Global transaction account network information:** Phishing accounts and legitimate accounts exhibit significant network structural differences. Legitimate accounts typically have forming sparsely connected network structures with minimal clustering. Phishing accounts, often create high-density subnetworks. These structural anomalies (tightly connected node clusters, localized high connectivity, sudden interaction spikes) serve as important indicators for identifying phishing behavior.

Integrating useful information is a highly promising direction to address the above issues [9]. In this study, we propose a deep learning framework with multimodal fusion

Zhang Sheng is with Hefei University of Technology (e-mail: dcszhang@foxmail.com).

Liangliang Song is with Xidian University (e-mail: songliangl@stu.xidian.edu.cn).

Yanbin Wang is with Xidian University (e-mail: wangyanbin15@mails.ucas.ac.cn). Yanbin Wang is the corresponding author.

*These authors contributed equally to this work.

†Corresponding author.

for fraud detection in blockchain transaction data. Compared with traditional methods, the proposed approach effectively captures both global structural relationships in transaction networks and local semantic patterns embedded in transaction records, achieving higher accuracy and robustness in detecting complex fraud behaviors.

Specifically, we first construct a global account interaction graph to represent the relationships between blockchain transaction accounts. Each node in the graph corresponds to an account, while the edges capture the transaction behaviors, such as frequency, transaction value, and temporal patterns. To extract meaningful structural features from this graph, we employ graph-based representation learning, which aggregates information from neighboring accounts to capture both direct and indirect relationships within the transaction network. This step enables the model to uncover global interaction patterns that are indicative of fraudulent behaviors.

Simultaneously, we process the semantic information embedded in transaction data using a pre-trained text representation model. The model converts textual descriptions, such as transaction amounts, smart contract details, and other metadata, into high-dimensional feature vectors. This process allows the model to identify local contextual relationships, such as recurring transaction patterns or anomalous textual characteristics associated with suspicious accounts.

To effectively leverage both structural and semantic insights, we propose a dynamic feature fusion mechanism that adaptively integrates these two feature spaces. The mechanism learns to balance global network structures and local transaction semantics based on their relative importance for each transaction, enabling the model to detect subtle and complex fraud patterns with high accuracy.

By combining these complementary perspectives—global structural relationships and local semantic features—our approach significantly improves the robustness and precision of fraud detection. Experimental results on real-world blockchain datasets demonstrate that the proposed ETH-GBERT model achieves state-of-the-art performance. Specifically, on the Multigraph dataset, the model achieved an F1 score of 94.71%, significantly outperforming the best-performing baseline (Role2Vec, F1 score of 74.13%), with an improvement of 20.58%. On the Transaction Network dataset, ETH-GBERT achieved an F1 score of 86.16%, representing a substantial enhancement over the next best model (Role2Vec, F1 score of 71.39%) , improving by 14.77%. Similarly, on the B4E dataset, ETH-GBERT obtained an F1 score of 89.79%, surpassing the highest baseline performance (Role2Vec, F1 score of 74.25%) by 15.54%. Additionally, the proposed model demonstrated superior recall (89.57%) and precision (90.84%), further highlighting its robustness in identifying phishing accounts. These results highlight the model's effectiveness in capturing complex fraud patterns, its robustness in handling imbalanced data distributions, and its ability to integrate structural and semantic features dynamically.

The main contributions of this study are as follows:

1) A dynamic multimodal fusion model is proposed, which innovatively combines graph structure information with text semantic similarity information to enhance the fraud detection performance in blockchain smart contracts.
2) A complete set of data processing flow is developed, including the extraction of transaction data, the generation of adjacency matrix, and the processing of text representation based on BERT [10], which provides a useful reference for other blockchain applications.
3) The effectiveness of the proposed method is verified through experiments, and the results show that the method performs well in detecting complex frauds and significantly outperforms existing benchmark models.

II. Related Work

In recent years, with the rapid development of blockchain technology, the frequent occurrence of fraud in blockchain networks has become a global challenge. Researchers and developers have developed various fraud detection methods to address these challenges and ensure the security and reliability of blockchain systems [11]. This section reviews existing phishing fraud methods, focusing on their technological innovations and limitations.

A. Graph-based Fraud Detection

In blockchain networks, transaction data usually has a complex relational structure, and graph-based models can effectively capture these complex relationships and excel in fraud detection. Especially in blockchain platforms like Ethereum, Graph Neural Networks (GNNs) are widely used to detect fraud. For example, Tan [12] proposed a model based on Graph Convolutional Networks (GCNs) for detecting fraud from Ethereum transaction records. They classified addresses as legitimate or fraudulent by constructing a transaction network and extracting node features. In addition, Kanezashi [13] investigated the application of Heterogeneous GNNs in Ethereum transaction networks, focusing on handling large-scale networks and the label imbalance problem. Li [14] also proposed a phishing detection framework called PDGNN, based on the Chebyshev-GCN, which can detect fraud in Ethereum transaction networks by extracting transaction subgraphs and training a classification model, effectively distinguishing normal accounts from phishing accounts in large-scale Ethereum networks. Wang [15] proposed the Transaction SubGraph Network (TSGN) framework to enhance phishing detection in Ethereum by constructing transaction subgraphs that capture essential features of transaction flows. Hou [16] proposed an Ethereum phishing detection method based on GCN and Conditional Random Field (CRF). This method first utilizes DeepWalk to generate initial features for each account node in the transaction graph, then employs GCN to learn graph-structured representations, capturing the transactional relationships between accounts. To enhance classification performance, a CRF layer further encourages similar nodes to learn similar representations.

B. Fraud Detection Based on Time Series Data

Time series data analysis plays an important role in blockchain fraud detection, especially in processing transaction records and detecting abnormal behaviours. Ethereum, as

one of the major blockchain platforms, contains a large amount of time-series information, such as transaction time, frequency, and value fluctuations, which can be used to identify potential fraudulent behaviours. Hu [17] investigated the application of time-series analysis methods based on the Long Short-Term Memory (LSTM) network in Ethereum smart contracts. Another study by Farrugia [18] proposed the use of the XGBoost model combined with time series features for illegal account detection in Ethereum. The study highlighted the importance of time series features, such as time intervals, in identifying illegal accounts by extracting key time series features and combining them with a machine learning model. Pan [19] proposed a system called EtherShield, which combines time interval analysis and contract code features to detect malicious behaviour on the Ethereum blockchain.

C. Hybrid Methods

Hybrid methods integrate various types of information, such as graph data, time-series data, and semantic information, achieving higher detection accuracy and robustness, effectively identifying complex and dynamic fraud patterns in Ethereum malicious transaction detection [20]. Li [21] proposed the Temporal Transaction Aggregation Graph Network (TTAGN) for Ethereum phishing detection, utilizing temporal transaction data to improve accuracy. TTAGN combines temporal edge representation, edge-to-node aggregation, and structural enhancement to capture transaction patterns and network structure, outperforming existing methods on real-world datasets. Wen [22] proposed a hybrid feature fusion model named LBPS for phishing detection on Ethereum, combining LSTM-FCN and BP neural networks. This model integrates features extracted through manual feature engineering and transaction records analysis, using BP neural networks to capture hidden relationships between features and LSTM-FCN networks to extract temporal features from transaction data. Chen [23] proposed the DA-HGNN model, a hybrid graph neural network with data augmentation for Ethereum phishing detection. This model utilizes data augmentation to address sample imbalance, integrates Conv1D and GRU-MHA to extract temporal features, and employs SAGEConv to capture structural features from the transaction graph.

Compared with the models reviewed above, our proposed ETH-GBERT model provides distinct advantages. Unlike purely graph-based models (such as GCN [7] or GAT [24]), ETH-GBERT incorporates rich textual transaction semantics, enabling it to detect phishing accounts that might not form clear structural patterns. Meanwhile, compared with text-only models such as BERT4ETH [25], our model leverages the global structural information provided by the graph embeddings to enhance detection of complex interaction patterns not identifiable through semantics alone. Although existing hybrid approaches (e.g., TTAGN [21], LBPS [22]) also combine multiple feature types, ETH-GBERT uniquely employs a dynamic fusion mechanism, adaptively weighting semantic and structural information depending on input complexity, which significantly improves robustness and flexibility in heterogeneous blockchain environments.

III. Methodology

In this chapter, we describe in detail a dynamic multimodal fusion approach for blockchain transaction data fraud detection. The proposed method integrates graph-based representation learning to capture global relationships within transaction networks and semantic feature extraction to identify local contextual patterns from transaction records. By leveraging a dynamic feature fusion mechanism, the model effectively combines structural and semantic information to enhance its ability to detect complex fraud behaviors, as illustrated in Figure 1. This chapter includes the detailed steps of our approach, starting with data generation and preprocessing, followed by a comprehensive explanation of the model architecture and the training process used to optimize performance.

A. Data generation and pre-processing

In the processing of blockchain transaction datasets, each transaction record typically contains several fields, such as `tag`, `from_address` (sender address), `to_address` (recipient address), `value` (transaction value), and `timestamp` (transaction timestamp). These fields describe the transaction behavior, the time it occurred, and the parties involved. To more effectively analyze and model transaction relationships, we need to properly classify and reorganize the transaction data.

Specifically, we classify all transaction data by sender and recipient addresses, constructing a transaction record structure based on accounts. This classification step not only simplifies transaction storage and access but also lays the foundation for subsequent graph structure construction and local semantic analysis.

Each transaction contains two account addresses, the sender (`from_address`) and the recipient (`to_address`). We classify transactions based on the sender's address (`from_address`), treating it as the transaction record of an account. Each transaction is labeled as an "outgoing" transaction, with the field `in_out` = 1. Similarly, when an account is the recipient, the transaction is labeled as an "incoming" transaction, with the field `in_out` = 0.

The classified transaction records are stored in a dictionary `accounts`, where the keys are account addresses and the values are lists of all transaction records for that account. Each list associated with an account contains all outgoing and incoming transactions related to that account. In this way, by separating and indexing transaction records by account, we can quickly retrieve the transaction history of any account, especially when analyzing account behavior patterns or transaction frequency.

1) Time Aggregation Feature Enhancement: To improve the information expression capability of transaction data in the time dimension, we particularly focus on the time aggregation characteristics of transactions during the data generation and preprocessing stages. By enhancing the time aggregation features, we can effectively capture some potential abnormal account behaviors, especially those accounts that engage in a large number of fund transactions within a short period [26]. These behaviors are often typical characteristics of phishing accounts, so analyzing and utilizing information in the time

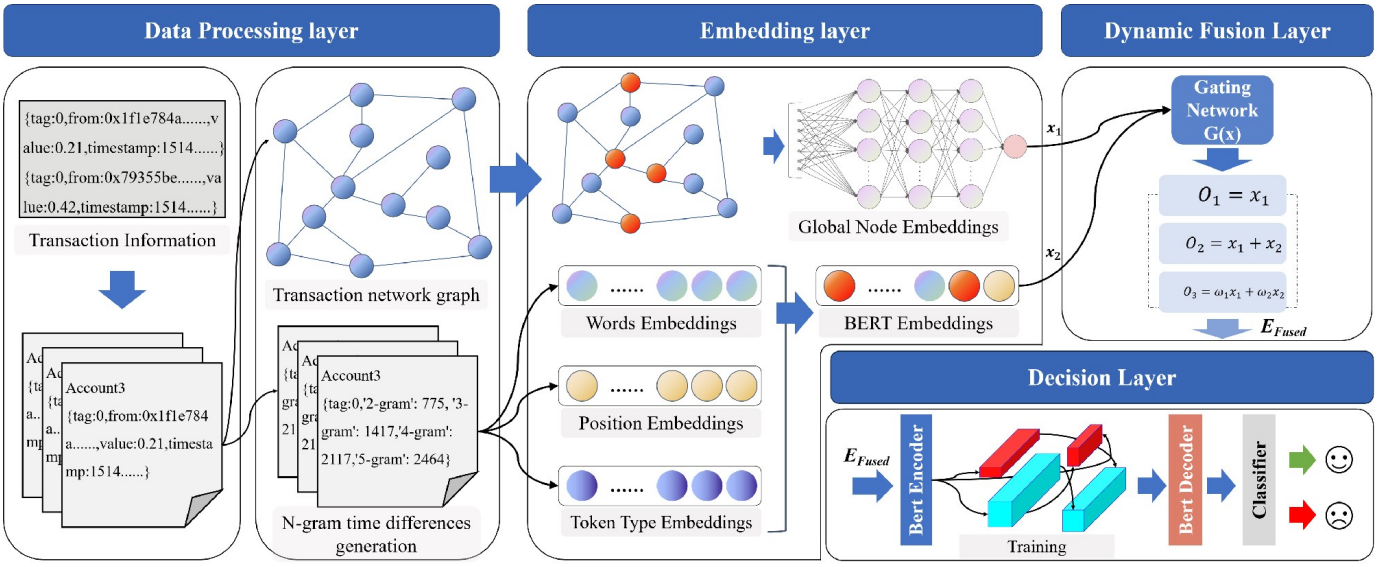

Fig. 1. Architecture of the Dynamic Feature Fusion Model for Blockchain Phishing Detection.
(Note: This figure has been revised in response to reviewer comments.)

dimension is crucial for accurately detecting fraudulent activities.

When processing the transaction data of each account, we first sort the transaction records based on the timestamp. The purpose of sorting is to ensure that the subsequent time difference calculations reflect the actual order of the transactions, providing foundational support for time aggregation features. By sorting the transactions in chronological order, we can capture the flow of funds in an account over a specific period and further analyze the frequency and density of its transaction behavior.

To quantify the degree of frequent transactions in a short time, we introduce the n-gram time difference feature. Specifically, the n-gram time difference measures the compactness of transaction times by calculating the time difference between a transaction and the previous $n-1$ transactions. We calculate the time differences for 2-gram to 5-gram, with the formula as follows:

$$\Delta T_n = T_i - T_{i-(n-1)}$$

where t_i denotes the timestamp of the ith transaction and $t_{i-(n-1)}$ denotes the timestamp of the $i-(n-1)$th transaction for the account. If the number of transactions is not sufficient to calculate the n-gram, the time difference is set to 0.

The n-gram time difference feature allows us to capture patterns of frequent trading over short periods. For example, if an account makes multiple inbound and outbound trades within a few minutes, the n-gram time difference will be significantly smaller, and this temporal aggregation reflects the account's high frequency of trades over a short period, which is often closely associated with phishing behavior.

2) Graph Data Generation: To effectively capture the inter-account relationships in blockchain transaction data, we first construct a graph-based data structure to represent the transaction network. In this section, we use an adjacency matrix $\mathbf{A}$ to quantify the connection weights between accounts in the transaction network. The process of generating this graph representation involves the following steps:

1) **Creating the Zero Matrix**
We first create a $n \times n$ zero matrix $\mathbf{A}$, where n denotes the number of unique account addresses. This adjacency matrix is used to store the connection weights between different accounts. The elements of the matrix $A[i,j]$ denote the transaction weights between account i and account j.

$$\mathbf{A} = \mathbf{0}_{n \times n}$$

2) **Traversing Transaction Records**
In order to populate the elements of the adjacency matrix, we need to iterate through all the transaction records T_k, where each transaction T_k contains a sender from_address$_k$ and a receiver to_address$_k$. We use an "address_to_index" dictionary to map these account addresses to indices in the adjacency matrix.

- The sender address is mapped as from_idx
- The receiver address is mapped as to_idx

The formulaic representation is as follows:

$$\text{from\_idx} = \text{address\_to\_index}(\text{from\_address}_k)$$

$$\text{to\_idx} = \text{address\_to\_index}(\text{to\_address}_k)$$

3) **Calculating Transaction Weights**

The weight of each transaction w_k reflects both the transaction value and the temporal characteristics of the transaction behavior. To effectively capture temporal features, we propose a weight calculation method based on the n-gram time differences. Specifically, the weight of a

transaction w_k is computed as a weighted sum of the n-gram time differences (ΔT_n). The formula for calculating the weights is defined as follows:

$$w_k = \text{value}_k \times \left(\sum_{n=1}^{N} \alpha_n \cdot \Delta T_n \right) \tag{1}$$

where:

- ΔT_n denotes the n-gram time difference as previously defined, i.e., $\Delta T_n = T_i - T_{i-(n-1)}$, representing the time interval between the i-th transaction and the $(i-(n-1))$-th transaction.
- α_n represents the weighting coefficients corresponding to different n-gram time differences. In our experiments, we empirically set these weights as inversely proportional to the n-gram order to emphasize shorter-term transaction bursts:

$$\alpha_n = \frac{1/n}{\sum_{j=1}^{N}(1/j)} \tag{2}$$

where N is the maximum n-gram considered (in our experiments, $N = 5$).

In addition, the transaction value value_k is also an important component of the weights, which we combine with the n-gram time difference to further adjust the weights of the transactions:

$$w_k = \text{value}_k \cdot \left(\sum_{n=1}^{N} \alpha_n \cdot \Delta t_{n,k} \right) \tag{3}$$

4) **Populating the Adjacency Matrix**

Once the weights of the transactions w_k are computed, they are accumulated to the corresponding positions in the adjacency matrix $A[\text{from\_idx}, \text{to\_idx}]$. Specifically, if multiple transactions occur between the same pair of accounts, their weights are summed up. This accumulation can be mathematically expressed as:

$$A[\text{from\_idx}, \text{to\_idx}] = \sum_{k \in \mathcal{T}(from\_idx, to\_idx)} w_k$$

where $\mathcal{T}(from\_idx, to\_idx)$ denotes the set of all transactions from account $from\_idx$ to account to_idx. Thus, the adjacency matrix entry reflects both the total frequency and aggregate transaction values between the two accounts.

This operation ensures that when multiple transactions occur between two accounts, the corresponding weights are accumulated in the appropriate elements of the adjacency matrix. This accumulation process effectively reflects both the frequency of transactions and the aggregate transaction values between accounts. The resulting adjacency matrix $\mathbf{A}$ serves as the input for graph-based representation learning, enabling the model to capture and analyze the global structural relationships within the transaction network.

In practice, the "address_to_index" dictionary size depends on the scale of the blockchain dataset under analysis, typically ranging from tens of thousands to millions of unique account addresses, especially when dealing with large blockchain networks like Ethereum. When new accounts emerge in real-time, they can be incrementally assigned new indices and appended to this dictionary. Consequently, the adjacency matrix needs to be dynamically updated by expanding its dimensions to accommodate these new accounts and their transactions. However, such dynamic updates may pose challenges in computational efficiency, as frequently resizing large adjacency matrices can be resource-intensive. Therefore, the proposed method, in its current form, primarily targets offline or batch analysis scenarios. For real-time phishing detection, additional optimization strategies, such as incremental graph updates, approximate adjacency structures, or streaming graph techniques, would be necessary.

3) Text Transaction Data Generation: In the transaction records of each account, the from_address, to_address and timestamp fields record the address information and timestamp of the account. Although these fields are important for transaction classification and temporal feature enhancement, they are not needed in text analysis, so we delete these fields before generating text data to simplify the data structure and retain key information such as transaction value and label.

Recent studies have shown that Transformer-based models, such as BERT, can also benefit from training with randomly or arbitrarily ordered sequences [27]. We take advantage of this property by randomly rearranging the transaction list for each account. This operation disrupts the backward and forward order of transactions, allowing the model to focus on the content features of transactions rather than time-dependent information, thus avoiding possible noise interference.

For example, the list of trades for account A is $[T_1, T_2, T_3]$ before disruption, and after random disruption may become$[T_2, T_1, T_3]$.

Next, we tag each account with an overall tag. An account is labelled as fraudulent whenever there is a transaction in the account with tag = 1, i.e., the account is labelled with a tag of 1. This tag is given to the first transaction record of the account. To simplify the transaction logging, the tag information for the rest of the transactions is deleted and only the tag of the first transaction is retained. This is because even if only one transaction in the account is related to fraud, the account itself may be potentially risky and may even be used for wider fraudulent activity. Typically, phishing accounts tend to mask their malicious behaviour by disguising multiple normal transactions. Therefore, in order to ensure the security and effectiveness of fraud detection, we have adopted more stringent criteria to ensure that the model can identify potentially high-risk accounts and prevent them from engaging in further illegal transactions. This labelling approach can help the model learn the risk characteristics of the accounts more accurately and improve the overall detection effectiveness.

When generating text data, we process the transaction records of each account and convert them into a single line of descriptive text. The key fields of each transaction (e.g., label tag, transaction value, etc.) are combined to create a compact textual representation that encapsulates the transaction information for the corresponding account. This step produces the raw text corpus, which serves as input for subsequent semantic

TABLE I
FEATURE EXTRACTION AND TSV REPRESENTATION SUMMARY

Field	Description	Example Value
tag	Phishing label (1) / legitimate (0)	1
value	Transaction amount transferred	5.06854256
in_out	Transaction direction (1:out, 0:in)	1
2-gram	Time delta: current vs. t-1 (sec)	30 (seconds)
3-gram	Time delta: current vs. t-2 (sec)	90 (seconds)
4-gram	Time delta: current vs. t-3 (sec)	120 (seconds)
5-gram	Time delta: current vs. t-4 (sec)	300 (seconds)
TSV format example:		
tag=1, value=5.0685, in_out=1, 2-gram=30, ...		

feature extraction through a pre-trained text representation model. The format can be clearly illustrated by a concrete example:

Phishing Account Example: `tag=1, value=5.0685, in_out=1, 2-gram:30, 3-gram:60, 4-gram:90, 5-gram:120; value=3.7451, in_out=0, 2-gram:30, 3-gram:60, 4-gram:90, 5-gram:120;`

Normal Account Example: `tag=0, value=0.0340, in_out=1, 2-gram:0, 3-gram:0, 4-gram:0, 5-gram:0;`

In these examples, the initial tag number represents the account-level label (`1` for phishing, `0` for legitimate), while the subsequent data represent transaction records presented in a randomly permuted order, where each account-level instance may encompass multiple individual transactions. This simplified representation allows the model to learn semantic patterns from transaction values without overfitting to temporal order or position-specific biases.

The generated textual transaction dataset is partitioned in the ratio of 80% training set, 10% validation set, and 10% test set. This data partitioning ensures that the model can learn enough features during the training process as well as perform performance tuning with the validation set, while verifying the model's generalisation ability on the test set.

4) Summary of Extracted Features and TSV Representation: To clearly illustrate the features extracted and included in the final transaction representation used in our experiments, we provide a detailed summary in Table I. Each row in the final TSV file corresponds to one blockchain account, with transaction information concatenated as textual descriptions.

This explicit representation facilitates semantic modeling using Transformer-based approaches, as transactions are encoded as textual sequences reflecting both their numeric and temporal attributes.

5) Text Data Cleaning: After generating the textual transaction data, further pre-processing steps are applied to ensure compatibility with the input format required for the downstream semantic representation model. These steps include reading the generated TSV files, tokenizing the text into subword units, and transforming it into a format suitable for deep learning-based training.

We first read the generated train.tsv and dev.tsv files, which contain the processed training set and validation set data. To ensure that the models are exposed to diverse data distributions during training, we randomly disrupt the data order to avoid overfitting the models to a specific data order. In addition, the test set data was read from test.tsv and similarly randomly disrupted.

After reading and shuffling the data, the training, validation, and test sets were combined into a unified data frame. From this, two key columns were extracted: the transaction text description (corpus) and the account label (y). The transaction text description captures the account's transaction behavior, while the label indicates whether the account is associated with fraudulent activity. This operation produces the input corpus and the corresponding supervisory signals (labels) required for subsequent semantic feature extraction and model training.

The textual corpus is then tokenized into subword units using BERT's WordPiece tokenizer. During this tokenization process, tokens are normalized by converting all characters to lowercase and applying standard Unicode normalization (NFKC), following the original BERT preprocessing recommendations [10]. This normalization ensures consistent token representations, reducing vocabulary redundancy and improving model efficiency. Subsequently, tokenized sequences are converted into token IDs, which serve as inputs to the embedding layer of the text processing model for subsequent training. To ensure robustness, the order of documents is intentionally shuffled, exposing the model to unordered and varied inputs during training. Additionally, the labeled data y is aligned with the tokenized sentences and used as supervisory signals for the supervised learning process.

This was followed by a tokenization process to segment each document into a series of tokens (sub-word units), which were then normalized and encoded as necessary. This step ensures that the transaction text is transformed into a format suitable for semantic representation models, resulting in sequences of token IDs. These token IDs serve as inputs to the embedding layer of the text processing model for subsequent training. To ensure robustness, the order of documents is intentionally shuffled, exposing the model to unordered and varied inputs during training. Additionally, the labeled data y is aligned with the tokenized sentences and used as supervisory signals for the supervised learning process.

The dataset generated in the above steps contains global transactional relationships and local transactional semantic information, providing multimodal input for subsequent model training.

B. ETH-GBERT Model Architecture

To address the challenge of detecting complex fraudulent activities in blockchain transactions, we propose the ETH-GBERT Model, a deep learning framework designed to simultaneously capture global structural relationships and local semantic similarities. While transaction networks contain rich global patterns that reflect account interactions, transaction records hold local contextual details that can signal fraudulent behaviors. Existing methods often focus on one aspect, failing to leverage the complementary strengths of both.

In this study, we adopt Graph Convolutional Networks (GCNs) to capture the global transaction relationships embedded in account interaction graphs. GCNs are particularly suited

for extracting structural features from graph-based data, making them ideal for modeling the relationships in blockchain transaction networks. Simultaneously, we use a pre-trained BERT model to analyze the local semantic features present in transaction text data, effectively capturing the contextual meaning and subtle patterns in transaction details.

By integrating these two components through a multimodal fusion mechanism, the ETH-GBERT Model combines insights from both global structural features and local semantic representations to enhance fraud detection performance. The following sections provide a detailed explanation of the architecture and design of the ETH-GBERT Model components.

1) Model Architecture: The ETH-GBERT model integrates two core modules: a GCN module for transaction account graphs and a BERT module for textual transaction data. Specifically, we use the following architectural configurations:

- **BERT component:** Pre-trained BERT-base model comprising 12 transformer encoder layers, with a hidden size of 768 and 12 attention heads.
- **GCN component:** A two-layer Graph Convolutional Network, with each layer having a hidden dimension size of 128.
- **Gating network:** The architecture employs a two-layer multilayer perceptron (MLP) with a hidden dimensionality of 128 and ReLU activation, which adaptively generates a probability vector to determine the relative contribution weights of each perspective within the fused multimodal embedding representation.

The overall model structure can be divided into the following parts:

i. **Graph-Based Representation Module**: Primarily captures global relationships within the transaction network. Through the GCN layers, the relationships between transaction accounts are convoluted, generating node embeddings (account embeddings) with global semantic information.
ii. **Semantic Feature Extraction Module**: Extracts local semantic information from transaction text data. The BERT model deeply represents the transaction records for each account and generates high-dimensional text embeddings.
iii. **Multimodal Fusion**: The GCN-generated global account embeddings and BERT-produced local text embeddings are fused, forming a multimodal embed vector. This fusion enables the model to take advantage of both the transaction network structure and the text features for fraud detection.
iv. **Classifier**: The fused embedding vector is passed through a fully connected layer for classification, outputting predictions to determine whether the account is related to fraudulent behavior.

2) Graph-Based Representation Module Design: Adjacency Matrix Input. The input to the GCN module is the adjacency matrix $\mathbf{A}$ of the transaction account graph, where the element $A[i, j]$ represents the transaction weight between the account i and the account j. This adjacency matrix is obtained from the graph data generation steps described earlier, incorporating transaction amounts and time features.

Graph Convolution Layer (GCN Layer). In the GCN module [7], the transaction account graph undergoes feature extraction through multiple graph convolution layers. The convolution operation in each layer is represented by the following formula:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{4}$$

where:

- $\mathbf{H}^{(l)}$ represents the node feature matrix at the l-th layer (account embedding matrix), and the initial $\mathbf{H}^{(0)}$ is the initial feature of the transaction accounts;
- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops;
- $\tilde{\mathbf{D}}$ is the degree matrix of the adjacency matrix;
- $\mathbf{W}^{(l)}$ is the weight matrix at the l-th layer;
- σ is a non-linear activation function, such as ReLU.

Through multiple convolution operations, the model aggregates the global information of the transaction network layer by layer, eventually generating node embeddings with global transaction relationships.

3) Semantic Feature Extraction Module Design: **Text Input and Initial Embeddings.** The input to the BERT module is the transaction text data. After being cleaned and tokenized, the text data is converted into token sequences. These token sequences are embedded using BERT's Word Embedding, Position Embedding, and Token Type Embedding layers [10]:

$$\mathbf{E}_{\text{BERT}} = \mathbf{E}_{\text{word}} + \mathbf{E}_{\text{position}} + \mathbf{E}_{\text{token\_type}} \tag{5}$$

Fusion with Graph Embeddings. Before being processed by the Transformer encoder, the embeddings from BERT ($\mathbf{E}_{\text{BERT}}$) are dynamically fused with graph-based embeddings to produce fused embeddings ($\mathbf{E}_{\text{Fused}}$). The detailed fusion mechanism and its adaptive weighting strategy are elaborated in the next subsection (III-B4).

BERT Encoding Layer. The fused embeddings $\mathbf{E}_{\text{Fused}}$ are then passed through BERT's multi-layer Transformer encoder, generating higher-level representations. Formally, this encoding step is defined as:

$$\mathbf{H}_{\text{fusion}} = \text{TransformerEncoder}(\mathbf{E}_{\text{Fused}}) \tag{6}$$

The resulting $\mathbf{H}_{\text{fusion}}$ serves as input to the final classification module.

4) Multimodal Fusion: In the multimodal fusion stage of the model, we introduce a **dynamic feature fusion mechanism** inspired by DynMM [28], which adaptively determines the contributions of BERT and GCN embeddings for each input instance.

Fusion Strategy. Our approach employs a **gating network** $G(x)$ to generate instance-specific fusion weights. This allows the model to dynamically decide how much information to extract from the existing embeddings. Specifically, three fusion strategies are considered:

- **BERT-only embeddings** E_{BERT}: Using textual information exclusively for prediction.

- **GCN-enhanced BERT embeddings** $E_{\text{GCN\_Enhanced}}$: GCN embeddings that integrate structural graph information and are enhanced with contextual features from BERT.
- **A weighted combination of BERT and GCN embeddings**:

$$E_{\text{Fusion}} = \alpha \cdot E_{\text{BERT}} + (1 - \alpha) \cdot E_{\text{GCN\_Enhanced}} \quad (7)$$

where α is a learnable parameter initialized to 0.5.

Dynamic Weight Calculation. The gating network $G(x)$ takes as input the concatenated features $[\mathbf{E}_{\text{BERT}}, \mathbf{E}_{\text{GCN\_Enhanced}}]$ and outputs fusion weights $g = [g_1, g_2, g_3]$ corresponding to the three fusion strategies:

$$g_i = \frac{\exp\left((\log G(x)_i + b_i)/\tau\right)}{\sum_{j=1}^{3} \exp\left((\log G(x)_j + b_j)/\tau\right)}, \quad i \in \{1, 2, 3\} \quad (8)$$

where $b_i \sim \text{Gumbel}(0, 1)$ is Gumbel noise, and τ is the temperature parameter controlling the sharpness of the resulting probability distribution. Specifically, when τ is large, the output distribution becomes smoother and approaches a uniform distribution, resulting in more balanced or equal weighting across the three fusion strategies. Conversely, as τ decreases, the distribution sharpens, eventually approaching a one-hot distribution that strongly favors a single fusion strategy. In practice, we adjust τ to find an optimal balance between exploration (balanced fusion) and exploitation (selective fusion), enhancing the adaptability of our dynamic fusion mechanism.

To handle varying task complexities and data characteristics, the gating network $G(x)$ can be implemented using different architectures, such as Multi-Layer Perceptrons (MLPs), Transformer layers, or convolutional networks.

In this work, we implement the gating network as a **Multi-Layer Perceptron (MLP)**, consisting of two fully connected layers with a ReLU activation function.

The final fused embedding E_{Fused} is obtained as:

$$E_{\text{Fused}} = g_1 \cdot E_{\text{BERT}} + g_2 \cdot E_{\text{GCN\_Enhanced}} + g_3 \cdot E_{\text{Fusion}} \quad (9)$$

Adaptive Fusion Mechanism. This dynamic fusion mechanism enables the model to adapt its computational resources and fusion strategy based on the input complexity:

- For **easy inputs**, the gating network assigns higher weights to simpler strategies such as E_{BERT} or $E_{\text{GCN\_Enhanced}}$, reducing computational costs.
- For **complex inputs**, the gating network increases the contribution of the weighted combination E_{Fusion}, allowing the model to effectively integrate information from both modalities.

Although the fusion mechanism introduces additional computational costs, our experiments demonstrate that the training time remains manageable. For instance, when early stopping is disabled, the ETH-GBERT model requires approximately 19 minutes per epoch—totaling 12.5 hours (754 minutes) for 40 epochs. Notably, the model reached its peak validation weighted F1-score (94.565%) by the 4th epoch as shown in Figure 2, after which performance metrics stabilized. Given the substantial performance gains—evidenced by significantly higher F1-scores compared to baseline methods—this computational expense is justifiable, particularly in scenarios where detection accuracy is paramount.

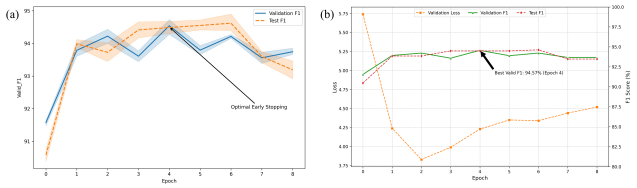

Fig. 2. Training Dynamics of ETH-GBERT with Early Stopping at Epoch 4. (a) The training loss curve shows that the model converges after 4 epochs; (b) The F1 score curve of the validation set reaches a peak at epoch 4.

In practice, the weights g_1, g_2, and g_3 are adaptively adjusted based on input complexity. For simpler, semantically focused transactions, the model may assign weights such as $[0.8, 0.1, 0.1]$, thereby favoring the BERT-based semantic embeddings. In contrast, structurally complex transactions involving multiple accounts might yield weights like $[0.2, 0.3, 0.5]$, emphasizing the hybrid embedding $\mathbf{E}_{\text{Fusion}}$.

Even though $\mathbf{E}_{\text{Fusion}}$ is already dynamically weighted, the additional gating mechanism—via g_1, g_2, and g_3—provides a higher-level adaptive decision layer. This extra flexibility allows the model to dynamically choose among single-modality embeddings (BERT or GCN-Enhanced) and the hybrid embedding, thereby enhancing its adaptability to heterogeneous blockchain data. Our experimental results confirm that this dynamic gating significantly contributes to the overall performance and flexibility of the model.

5) Classifier Design: The fused multimodal embedding vector $\mathbf{H}_{\text{fusion}}$ is input into a fully connected layer for the classification task. Through the Softmax layer, the model outputs the probability of whether an account is related to fraudulent behavior:

$$\mathbf{y} = \text{Softmax}(\mathbf{W}_{\text{fusion}}\mathbf{H}_{\text{fusion}} + \mathbf{b}_{\text{fusion}}) \quad (10)$$

where $\mathbf{W}_{\text{fusion}}$ and $\mathbf{b}_{\text{fusion}}$ are the weight matrix and bias vector of the classifier, respectively.

ETH-GBERT Model enhances the joint learning of global relationships and local semantic information in blockchain transactions through the fusion of GCN and BERT embeddings. Through multimodal fusion, the model improves its ability to detect complex fraudulent behaviors effectively.

IV. Validation

A. Dataset review

As shown in Table II, we evaluate our model on three blockchain fraud detection datasets with distinct characteristics. Detailed descriptions and key attributes are provided below.

1) Multigraph Dataset: This dataset is publicly available and is provided by Chen et al. (2021). The dataset is obtained by performing second-order breadth-first search (BFS) from known phishing nodes over a large-scale Ethernet transaction

TABLE II
COMPARATIVE SUMMARY OF DATASET ATTRIBUTES

Attribute	Multigraph	Transaction Network	B4E
Time Span	2015–2019	2017–2022	2017–2022
Nodes/Accounts	2,973,489	60,000	597,258
Edges/Transactions	13,551,303	200,000	1,678,901
Phishing Accounts	1,165	1,259	3,220

network. The dataset contains 2,973,489 nodes, 13,551,303 edges, and 1,165 phishing nodes. [29]

2) Transaction Network Dataset: This dataset was collected by Wu et al. (2022) through Ethernet nodes. It includes 1,259 phishing accounts and an equal number of normal accounts. The first-order neighbours of each account and the transaction edges between them are also included in the dataset, and the subnetwork contains about 60,000 nodes and 200,000 transaction edges [30].

3) BERT4ETH: We use the BERT4ETH dataset provided by Hu et al. (2023), which is generated from sequences of Ether transactions spanning from 2017 to 2022. The dataset contains 597,258 accounts, 1,678,901 transaction edges, and 3,220 labeled phishing accounts. It also includes de-anonymised data (ENS and Tornado Cash) and ERC-20 token logs. BERT4ETH captures multihop relationships between trading accounts and is suitable for phishing account detection and account de-anonymisation tasks. This dataset is an important component of our experiments and helps to further evaluate the performance of the model [25].

B. Baseline

In this experiment, we selected three common categories of baseline models for comparison:

1) Graph embedding methods based on random walks, including DeepWalk [31], Trans2Vec [30], Dif2Vec [32], and Role2Vec [33], [34];
2) Graph neural network(GNN) models, including GCN [7], GraphSAGE [35], and GAT [24];
3) BERT4ETH, a model designed specifically for fraud detection on Ethereum [25].

DeepWalk generates node sequences through random walks on the graph and employs the skip-gram model to learn low-dimensional representations of nodes. Trans2Vec builds on DeepWalk by incorporating transaction heterogeneity and temporal features, designed specifically for detecting phishing accounts in the Ethereum network. Dif2Vec adjusts the sampling probabilities of nodes during random walks to enhance the diversity of embeddings by increasing the sampling of low-degree nodes. Role2Vec learns structural roles of nodes rather than focusing solely on proximity relationships, generating more generalizable embeddings.

Regarding GNN-based models, GCN aggregates the features of neighboring nodes via convolution operations to learn node representations, making it suitable for tasks such as node classification. GraphSAGE generates new node embeddings by sampling and aggregating the features of neighboring nodes, which enables it to handle large-scale graph data. GAT introduces an attention mechanism, dynamically assigning weights to each node's neighbors to aggregate node information more effectively.

BERT4ETH is specifically designed for detecting fraudulent activities on the Ethereum network, leveraging BERT along with transaction data features from the Ethereum network to identify fraudulent behavior within blockchain transactions.

In our experiments, all baseline models, including BERT4ETH, DeepWalk, Trans2Vec, Dif2Vec, Role2Vec, GCN, GSAGE, and GAT, were implemented according to the original configurations specified in their respective papers. This ensures a fair comparison of performance across different models.

V. PREPROCESSING AND TRAINING SETTINGS

In this section, we describe the ETH-GBERT preprocessing setup, initial parameters, loss function, and evaluation metrics used in our experiment.

A. Data Loading and Preprocessing

Before training, the dataset was split into training, validation, and test sets, accounting for 80%, 10%, and 10% of the total data, respectively. We used PyTorch's `DataLoader` to load the data in mini-batches, with shuffling applied during the training process. The training set is used to update model parameters, the validation set evaluates the model's generalization ability, and the test set is used for final performance evaluation.

B. Hyperparameter Settings

The following hyperparameters were set during the model training:

- **Learning rate**: The initial learning rate was set to 8×10^{-6}, and a learning rate scheduler was employed to adjust the learning rate dynamically.
- **Regularization coefficient**: L2 regularization was applied with a coefficient of $\lambda = 0.001$ to prevent overfitting.
- **Batch size and gradient accumulation**: The batch size was set to 32. We adopted gradient accumulation to save memory, updating the model's parameters after every 2 mini-batches.
- **Epochs**: We have set the maximum number of epochs to 40. Prior work [23] indicates convergence within 30–50 epochs for similar tasks. Training our Ethereum subgraph requires significant memory (about 12.4GB per epoch), and 40 epochs guarantee stable training within 24 hours.

C. Loss Function and Optimizer

We used the cross-entropy loss function for the classification task [36], defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right) \qquad (11)$$

where N is the batch size, y_i is the ground truth label, and p_i is the predicted probability.

TABLE III
PERFORMANCE COMPARISON OF ETH-GBERT AND BASELINE MODELS ON VARIOUS DATASETS

Model	Multigraph			Transaction Network			B4E		
	F1 Score	Recall	Precision	F1 Score	Recall	Precision	F1 Score	Recall	Precision
BERT4ETH	67.11	61.25	74.21	64.21	62.17	66.39	64.26	63.58	64.95
DeepWalk	58.44	58.21	58.67	59.21	58.31	60.14	54.51	55.38	53.67
Trans2Vec	52.13	51.36	52.92	54.28	56.26	52.43	55.31	54.96	55.66
Dif2Vec	65.27	64.21	66.37	62.11	62.54	61.69	63.25	63.54	62.96
Role2Vec	74.13	74.52	73.74	71.39	71.58	71.20	74.25	74.25	74.25
GCN	42.29	74.07	29.59	41.12	73.37	28.56	64.71	72.68	58.31
GSAGE	35.47	34.77	36.20	33.79	32.99	34.64	53.28	60.47	47.62
GAT	39.98	79.82	26.67	41.61	77.56	28.43	61.50	85.20	48.12
ETH-GBERT	**94.71**	**94.71**	**94.71**	**86.16**	**87.82**	**84.56**	**89.79**	**89.57**	**90.84**

The AdamW optimizer was employed for optimization, combining the adaptive learning rate of Adam with L2 regularization through weight decay. The update rule for AdamW is given by:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{12}$$

where m_t and v_t are the first and second moments of the gradients, and ϵ is a small constant to avoid division by zero.

D. Evaluation Metrics

At the end of each epoch, the model's performance was evaluated on the validation set using precision, recall, and F1 score as evaluation metrics:

- **Precision**:
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
- **Recall**:
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
- **F1 Score**:
$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Here, TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively. These evaluation metrics provide a comprehensive view of the model's classification performance and help monitor the generalization ability throughout the training process.

VI. PERFORMANCE

To evaluate the effectiveness of our proposed ETH-GBERT model in detecting fraud in blockchain transaction data, we compared its performance with several baseline models, including BERT4ETH, DeepWalk, Trans2Vec, Dif2Vec, Role2Vec, GCN, GSAGE, and GAT. These models were applied to three different datasets: Multigraph, Transaction Network, and B4E. The comparison focuses on key metrics such as F1 score, recall, and precision, as shown in Table III.

A. Overview of Model Performance

From the experimental results, it is evident that ETH-GBERT significantly outperforms all baseline models across the datasets in terms of F1 score, recall, and precision.

- **On the Multigraph dataset**, ETH-GBERT achieves an F1 score of 94.71, approximately 10 points higher than GAT (84.35). This demonstrates ETH-GBERT's ability to effectively combine graph structures and semantic information for superior fraud detection performance.
- **On the Transaction Network dataset**, ETH-GBERT achieves an F1 score of 86.16, with a recall of 87.82 and precision of 84.56. Compared to GAT (F1 score of 83.27) and GCN (F1 score of 83.29), ETH-GBERT provides enhanced accuracy in capturing the complexities of transaction relationships.
- **On the B4E dataset**, ETH-GBERT achieves an F1 score of 89.79, surpassing all the baseline models. Notably, ETH-GBERT excels in recall, achieving 89.57, highlighting its sensitivity in identifying potential fraud cases.

B. Comparison with Baseline Models

Several key insights can be drawn from the comparison with baseline models:

1) **BERT4ETH**: While BERT4ETH demonstrates reasonable performance in extracting local semantic information, its F1 scores on both the Multigraph and Transaction Network datasets (67.11 and 64.21, respectively) are significantly lower than ETH-GBERT. This highlights the importance of incorporating global structure information, which BERT4ETH lacks.
2) **GCN and GSAGE**: GCN and GSAGE struggle to achieve competitive F1 scores, with GCN scoring 42.29 on the Multigraph dataset and 41.12 on the Transaction Network dataset. These models are effective in capturing global transaction relationships but lack the ability to integrate local semantic information, limiting their performance in fraud detection tasks.
3) **GAT**: The GAT model benefits from its self-attention mechanism, achieving a relatively higher recall (e.g., 79.82 on the Multigraph dataset). However, its F1 scores

TABLE IV
PERFORMANCE IMPROVEMENT ANALYSIS VIA MULTIMODAL DYNAMIC FUSION

Model	Multigraph			Transaction Network			B4E		
	F1 Score	Recall	Precision	F1 Score	Recall	Precision	F1 Score	Recall	Precision
BERT Only	90.10	90.07	90.15	80.87	78.12	83.82	85.19	83.05	87.44
Difference(%)	*-4.61*	*-4.64*	*-4.56*	*-5.29*	*-9.70*	*-0.74*	*-4.6*	*-6.52*	*-3.40*
GCN Only	42.29	74.07	29.59	41.12	73.37	28.56	64.71	72.68	58.31
Difference(%)	*-52.42*	*-20.64*	*-65.12*	*-45.04*	*-14.45*	*-56.00*	*-25.08*	*-16.89*	*-32.53*
Simple Combination	84.55	84.15	86.29	83.27	83.75	83.55	85.35	88.16	82.71
Difference(%)	*-10.16*	*-10.56*	*-8.42*	*-2.89*	*-4.07*	*-1.01*	*-4.44*	*-1.41*	*-8.13*
Weighted Combination	92.43	92.51	92.47	85.21	83.75	**86.73**	88.23	86.34	90.20
Difference(%)	*-2.28*	*-2.20*	*-2.24*	*-0.95*	*-4.07*	*+2.17*	*-1.56*	*-3.23*	*-0.64*
ETH_GBERT	**94.71**	**94.71**	**94.71**	**86.16**	**87.82**	84.56	**89.79**	**89.57**	**90.84**

TABLE V
PERFORMANCE WITH DIFFERENT NORMAL TO FRAUD RATIOS

Ratio	Multigraph			Transaction Network			B4E		
	F1 Score	Recall	Precision	F1 Score	Recall	Precision	F1 Score	Recall	Precision
1:9	78.50	80.10	77.90	75.20	76.90	74.50	70.30	72.20	69.80
2:8	81.30	82.40	80.20	77.80	79.50	76.70	73.10	74.80	72.90
3:7	83.70	84.50	82.90	80.30	81.80	79.90	75.40	77.20	74.60
4:6	87.50	88.20	86.70	84.10	85.40	83.82	79.10	80.70	78.90
5:5	**94.71**	**94.71**	**94.71**	**86.16**	**87.82**	**84.56**	**89.79**	**89.57**	**90.84**
6:4	89.30	90.20	88.70	83.80	85.10	82.90	81.18	82.60	79.80
7:3	85.60	86.50	84.80	80.90	82.30	79.10	77.20	78.80	76.50
8:2	82.30	83.40	81.60	77.60	79.10	76.40	73.90	75.50	73.20
9:1	80.10	81.20	79.30	75.40	76.83	74.60	71.30	72.80	70.50

remain low (39.98 on Multigraph and 41.61 on Transaction Network), due to its limited ability to model textual features and complex fraud patterns.

4) **ETH-GBERT**: Our proposed ETH-GBERT model significantly outperforms all baseline models across all datasets. It achieves the highest F1 scores of 94.71, 86.16, and 89.79 on the Multigraph, Transaction Network, and B4E datasets, respectively. This performance demonstrates the effectiveness of ETH-GBERT in dynamically fusing global transaction network information with local semantic features from transaction texts, enabling superior fraud detection capabilities.

C. Improvement Analysis via Multimodal Dynamic Fusion

Table IV illustrates the performance improvements brought by multimodal dynamic fusion, showcasing the different performances of **Unimodal Models**, **Static Fusion Methods**, and **Dynamic Fusion**.

- **Unimodal Models:** The BERT-only model achieves strong results in the Multigraph dataset (F1 Score = 90.10) due to its ability to model language-centric features. However, it performs poorly in Transaction Network and B4E datasets (F1 Scores = 80.87 and 85.19), indicating its limitations in graph-based tasks. Conversely, the GCN model, which relies solely on graph information, performs poorly across all datasets, showing its limited capacity to model textual features.
- **Static Fusion Methods:** The GCN-enhanced BERT approach combines semantic and graph features, but its fixed fusion mechanism prevents dynamic weight adjustment to fully leverage their respective advantages. As a result, the performance gains are limited, and on some datasets, the metrics even perform worse compared to using BERT alone.
- **Dynamic Fusion (ETH-GBERT):** The ETH-GBERT model, leveraging dynamic fusion, achieves the best performance across almost all datasets. It dynamically adjusts the contributions of BERT and GCN, resulting in F1 Scores of 94.71, 86.16, and 89.79 in Multigraph, Transaction Network, and B4E datasets, respectively. Compared to static fusion, it offers consistent improvements (e.g., +2.28 in Multigraph, +1.56 in B4E).

The results highlight the limitations of unimodal and static fusion methods in handling multimodal data. Dynamic fusion, as implemented in ETH-GBERT, effectively balances textual and graph-based features, achieving superior performance and adaptability across diverse tasks. This demonstrates its potential as a robust multimodal learning approach.

D. Impact of Normal to Fraud Ratio on Model Performance

In this subsection, we evaluate how varying the ratio of normal to fraud transactions in the dataset affects the performance of the ETH-GBERT model across three different datasets: Multigraph, Transaction Network, and B4E.

We trained the ETH-GBERT model on datasets with varying ratios of normal to fraud transactions, ranging from 1:9 to 9:1. The key evaluation metrics—F1 Score, Recall, and Precision—were tracked for each dataset to understand the impact of different data distributions on the model's performance.

Table V presents the performance metrics for each dataset under different normal-to-fraud ratios. The results demonstrate that the ETH-GBERT model performs optimally on a balanced dataset (5:5 ratio), achieving the highest F1 Score, Recall, and Precision. For example, in the **Multigraph** dataset, the model reaches an F1 Score of 94.71, while in the **Transaction Network** dataset, the highest F1 Score is 86.16. The **B4E** dataset also shows strong performance, with an F1 Score of 89.79 at the 5:5 ratio.

Although the performance declines as the data becomes imbalanced, the ETH-GBERT model remains robust. The overall performance decrease is more pronounced in datasets with more complex transaction patterns, such as the **B4E** dataset, where the interaction between normal and fraud transactions may contain more nuanced features.

These findings suggest that while the ETH-GBERT model can handle imbalanced datasets, a balanced ratio between normal and fraud transactions helps the model achieve its best performance.

E. Insights from Experimental Results

We further analyze the experimental results to gain deeper insights into the performance variations across different datasets and fusion strategies:

- On the **Multigraph dataset**, the performance of the BERT-only model was notably strong (F1 score = 90.10), closely approaching our proposed ETH-GBERT (F1 = 94.71). This indicates that textual transaction semantics alone contain highly discriminative signals for phishing detection on this dataset. However, on the **Transaction Network dataset**, the BERT-only model performed poorly (F1 score = 80.87) compared to ETH-GBERT (F1 = 86.16). The significant performance gap suggests that structural relationships captured by the GCN component are critical for detecting complex transaction-based phishing behaviors that textual embeddings alone cannot adequately represent.
- Regarding fusion strategies, simple combination methods showed relatively limited effectiveness on the **Multigraph dataset** (F1 = 84.55), considerably behind ETH-GBERT (F1 = 94.71). This indicates that the Multigraph dataset might require more sophisticated dynamic weighting to effectively leverage both modalities. In contrast, for the **Transaction Network and B4E datasets**, the differences between simple/weighted combinations (F1 scores around 83-86) and ETH-GBERT (F1 scores 86.16 and 89.79 respectively) were comparatively smaller. These findings suggest that the incremental performance gains of dynamic fusion become particularly pronounced on datasets with distinctive modality strengths or high structural-semantic heterogeneity.

VII. Limitations and Future Directions

Although the proposed ETH-GBERT model demonstrated significant improvements in phishing detection on blockchain transaction data, several limitations should be acknowledged clearly:

- **Generalization to Other Fraud Types:** Currently, our method and experiments specifically focus on phishing detection. Extending our approach to other types of blockchain-related fraud, such as Ponzi schemes, money laundering, or ransomware payment detection, would require careful re-examination and possibly additional domain-specific feature engineering. Future research could investigate how well the proposed dynamic multimodal fusion generalizes across various types of blockchain fraud.
- **Real-time Detection vs. Offline Analysis:** The current ETH-GBERT model, due to computational complexity in multimodal embedding fusion and adjacency matrix construction, primarily targets offline or batch-mode analysis scenarios. Implementing this approach for real-time detection poses additional computational challenges, such as incremental graph updating and real-time embedding inference. Future research should explore incremental learning and efficient real-time fusion mechanisms for live blockchain monitoring.

VIII. Conclusion

In this paper, we proposed a novel dynamic multimodal fusion model(ETH-GBERT) for fraud detection in blockchain transactions. By adaptively integrating global structural features from transaction networks and local semantic information from transaction texts, the model effectively addresses the limitations of existing methods, achieving a better balance between computational efficiency and representation learning power.

To support the proposed model, we developed a comprehensive data processing pipeline, including graph construction for capturing inter-account relationships and temporal feature extraction using n-gram time differences. This pipeline enables the model to simultaneously analyze global structural patterns and local contextual features embedded within transaction data. Furthermore, the dynamic fusion mechanism introduced in this work adaptively adjusts the contributions of structural and semantic features based on transaction context, enhancing the model's accuracy and robustness in detecting complex fraudulent activities.

Through extensive experiments on large-scale blockchain datasets, our model demonstrated significant improvements over existing benchmark methods, achieving the highest F1 scores across multiple evaluation scenarios.

The key contributions of this study are as follows:

- Proposing a multimodal fusion framework that dynamically integrates structural and semantic information to enhance blockchain fraud detection.
- Developing a robust and efficient data processing pipeline that captures both global transaction relationships and temporal behavioral patterns.

- Introducing a dynamic feature fusion mechanism that adaptively balances feature contributions, improving detection precision and efficiency across varied contexts.
- Demonstrating the effectiveness of the proposed approach through experiments, where it significantly outperformed state-of-the-art models on multiple real-world datasets.

REFERENCES

[1] A. Pal, C. K. Tiwari, A. Behl, Blockchain technology in financial services: a comprehensive review of the literature, Journal of Global Operations and Strategic Sourcing 14 (1) (2021) 61–80.

[2] M. Bhowmik, T. S. S. Chandana, B. Rudra, Comparative study of machine learning algorithms for fraud detection in blockchain, in: 2021 5th international conference on computing methodologies and communication (ICCMC), IEEE, 2021, pp. 539–541.

[3] Z. Wenhua, F. Qamar, T.-A. N. Abdali, R. Hassan, S. T. A. Jafri, Q. N. Nguyen, Blockchain technology: security issues, healthcare applications, challenges and future trends, Electronics 12 (3) (2023) 546.

[4] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, Y. Cao, A survey on blockchain technology: Evolution, architecture and security, Ieee Access 9 (2021) 61048–61073.

[5] J.-Y. Lai, J. Wang, Y.-H. Chiu, Evaluating blockchain technology for reducing supply chain risks, Information Systems and e-Business Management 19 (4) (2021) 1089–1111.

[6] I. Givargizov, Unstable financial and economic factors in the world and their influence on the development of blockchain technologies, International Humanitarian University Herald. Economics and Management (01 2023). doi:10.32782/2413-2675/2023-55-11.

[7] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[8] A. Ancelotti, C. Liason, Review of blockchain application with graph neural networks, graph convolutional networks and convolutional neural networks, arXiv preprint arXiv:2410.00875 (2024).

[9] S. Liu, B. Cui, W. Hou, A survey on blockchain abnormal transaction detection, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2023, pp. 211–225.

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.

[11] J. Osterrieder, S. Chan, J. Chu, Y. Zhang, B. H. Misheva, C. Mare, Enhancing security in blockchain networks: Anomalies, frauds, and advanced detection techniques, arXiv preprint arXiv:2402.11231 (2024).

[12] R. Tan, Q. Tan, P. Zhang, Z. Li, Graph neural network for ethereum fraud detection, in: 2021 IEEE international conference on big knowledge (ICBK), IEEE, 2021, pp. 78–85.

[13] H. Kanezashi, T. Suzumura, X. Liu, T. Hirofuchi, Ethereum fraud detection with heterogeneous graph neural networks, arXiv preprint arXiv:2203.12363 (2022).

[14] P. Li, Y. Xie, X. Xu, J. Zhou, Q. Xuan, Phishing fraud detection on ethereum using graph neural network, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2022, pp. 362–375.

[15] J. Wang, P. Chen, X. Xu, J. Wu, M. Shen, Q. Xuan, X. Yang, Tsgn: Transaction subgraph networks assisting phishing detection in ethereum, arXiv preprint arXiv:2208.12938 (2022).

[16] W. Hou, B. Cui, R. Li, Detecting phishing scams on ethereum using graph convolutional networks with conditional random field, in: 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), IEEE, 2022, pp. 1495–1500.

[17] T. Hu, X. Liu, T. Chen, X. Zhang, X. Huang, W. Niu, J. Lu, K. Zhou, Y. Liu, Transaction-based classification and detection approach for ethereum smart contract, Information Processing & Management 58 (2) (2021) 102462.

[18] S. Farrugia, J. Ellul, G. Azzopardi, Detection of illicit accounts over the ethereum blockchain, Expert Systems with Applications 150 (2020) 113318.

[19] B. Pan, N. Stakhanova, Z. Zhu, Ethershield: Time-interval analysis for detection of malicious behavior on ethereum, ACM Transactions on Internet Technology 21 (1) (2024) 1–30.

[20] J. Sun, Y. Jia, Y. Wang, Y. Tian, Z. Sheng, Ethereum fraud detection via joint transaction language model and graph representation learning, Information Fusion (2025) 103074.

[21] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, G. Xiong, Ttagn: Temporal transaction aggregation graph network for ethereum phishing scams detection, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 661–669.

[22] T. Wen, Y. Xiao, A. Wang, H. Wang, A novel hybrid feature fusion model for detecting phishing scam on ethereum using deep neural network, Expert Systems with Applications 211 (2023) 118463.

[23] Z. Chen, S.-Z. Liu, J. Huang, Y.-H. Xiu, H. Zhang, H.-X. Long, Ethereum phishing scam detection based on data augmentation method and hybrid graph neural network model, Sensors 24 (12) (2024) 4022.

[24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[25] S. Hu, Z. Zhang, B. Luo, S. Lu, B. He, L. Liu, Bert4eth: A pre-trained transformer for ethereum fraud detection, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 2189–2197.

[26] W. Chen, Z. Zheng, E. C.-H. Ngai, P. Zheng, Y. Zhou, Exploiting blockchain data to detect smart ponzi schemes on ethereum, IEEE Access 7 (2019) 37575–37586.

[27] M. Clarke, Arbitrary-order sampling and hand motion modeling with transformers, Ph.D. thesis, Open Access Te Herenga Waka-Victoria University of Wellington (2023).

[28] Z. Xue, R. Marculescu, Dynamic multimodal fusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 2575–2584.

[29] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, Z. Zheng, Phishing scams detection in ethereum transaction network, ACM Transactions on Internet Technology (TOIT) 21 (1) (2020) 1–16.

[30] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, Z. Zheng, Who are the phishers? phishing scam detection on ethereum via network embedding, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (2) (2020) 1156–1166.

[31] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[32] B. Rozemberczki, R. Sarkar, Fast sequence-based embedding with diffusion graphs, in: Complex Networks IX: Proceedings of the 9th Conference on Complex Networks CompleNet 2018 9, Springer, 2018, pp. 99–107.

[33] N. K. Ahmed, R. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, H. Eldardiry, Learning role-based graph embeddings, arXiv preprint arXiv:1802.02896 (2018).

[34] F. Béres, I. A. Seres, A. A. Benczúr, M. Quintyne-Collins, Blockchain is watching you: Profiling and deanonymizing ethereum users, in: 2021 IEEE international conference on decentralized applications and infrastructures (DAPPS), IEEE, 2021, pp. 69–78.

[35] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).

[36] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.