An Enhanced Temporal Graph Network with Retrieval-Augmented Graph for Dynamic Link Prediction in Cryptocurrency Transactions

1st Isaac Amankona Obiri*
School of Software
Quanzhou University of Information Engineering
Quanzhou, China
isaac@qzuie.edu.cn*

2nd Ansu Badjie
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
202214080108@std.uestc.edu.cn

3rd Abigail Akosua Addobea
School of Software
Quanzhou University of Information Engineering
Quanzhou, China
akusha@qzuie.edu.cn

Abstract*—Link prediction in cryptocurrency transaction networks is inherently challenging due to their large scale, temporal dynamics, and evolving interaction patterns between entities. Traditional static graph-based approaches often fail to capture these temporal dependencies and are limited in modeling missing or incomplete transactional information. This work presents an Enhanced Temporal Graph Network (ETGN) with a Retrieval-Augmented Graph (RAG) model for dynamic link prediction in cryptocurrency transactions. The proposed framework integrates a Multi-Graph LSTM (MGLSTM) with graph attention mechanisms to jointly model structural and temporal dependencies while incorporating a missing information prediction module to mitigate data sparsity and incompleteness. Node representations are enriched through deterministic feature extraction and centrality-driven labeling using affinity and anti-affinity scores derived from transaction graphs. To enhance generalization across time slices, a graph memory module is introduced, enabling the retrieval of historical edge embeddings and contextual fusion for improved prediction accuracy. The model performs edge-level classification to predict the likelihood of future transactional links. Experiments conducted on real-world Bitcoin and Ethereum transaction network data, organized into temporal slices, demonstrate that the proposed ETGN-RAG framework achieves strong performance in terms of AUC-ROC, precision, recall, and F1-score. The results highlight the effectiveness of combining temporal graph learning, attention-based aggregation, and retrieval-augmented memory for robust link prediction in dynamic cryptocurrency transaction networks.

***Index Terms*—Dynamic Graph, Temporal Graph Neural Network, Retrieval-Augmented Graph, Edge Classification, Transaction Network, Missing Information Prediction.**

I. Introduction

The rapid growth of decentralized digital payment platforms, particularly Bitcoin [1] and Ethereum [2], has led to the continuous generation of massive transaction networks. In these networks, nodes represent users or smart contracts, while edges denote value transfers occurring over time. Empirical studies have demonstrated that cryptocurrency transaction graphs exhibit complex structural evolution, heterogeneous connectivity patterns, and strong temporal dependencies [3], [4]. Understanding and predicting the formation of future transaction links is fundamental for network analysis, system optimization, capacity planning, and behavioral modeling in blockchain ecosystems.

Transaction networks are inherently dynamic. User activity levels, interaction preferences, and connectivity patterns evolve continuously, and modeling these systems as static graphs ignores temporal signals that are essential for predicting future links. Temporal dependencies often reveal recurring transaction behaviors, evolving community structures, and long-range interaction patterns that cannot be captured from isolated snapshots [18]. Consequently, dynamic link prediction can be formulated as an edge-level temporal forecasting problem, where the objective is to estimate the likelihood of future interactions between node pairs based on historical structural context and temporal evolution.

Early graph representation learning methods focus on static networks, including DeepWalk [5], SDNE [6], and GAT [7], which learn node embeddings from fixed topology. Similarity-based approaches exploit neighborhood overlap and path-based metrics, while machine learning methods integrate structural features with classifiers [9]–[13]. Although effective on static graphs, these methods fail to capture temporal dynamics. To address this, dynamic graph models such as EvolveGCN [15] and Tail-GNN [16] incorporate temporal evolution and improve learning for sparse nodes. Tensor decomposition methods further model dynamic weighted networks [17], while recent studies highlight joint temporal–structural learning [18]. However, most approaches focus on short-term dynamics and node-level representations, lacking support for long-range

dependencies and edge-centric prediction in large-scale transaction networks.

Motivated by dynamic link prediction challenges, we propose ETGN-RAG, a temporal graph network with retrieval-augmented memory. The framework combines attention-based GNNs with a multi-graph LSTM to capture structural and temporal dependencies. Node features derived from structural metrics are processed across temporal slices. A memory module retrieves and fuses historical edge embeddings to enhance long-term context. A lightweight edge predictor with imbalance-aware training enables scalable and accurate link prediction.

The main contributions of this paper are summarized as follows:

- We propose an Enhanced Temporal Graph Network with Retrieval-Augmented Graph memory (ETGN-RAG) for predicting future links in dynamic transaction networks. The framework jointly models structural dependencies, temporal evolution, and historical contextual information to accurately forecast evolving interactions in large-scale blockchain graphs.
- We design a Multi-Graph LSTM (MGLSTM) cell that embeds Graph Attention Networks (GAT) into recurrent gating mechanisms, enabling simultaneous learning of neighborhood-aware structural representations and temporal dependencies across graph snapshots.
- We introduce a graph memory module that stores historical edge embeddings and supports similarity-based retrieval of relevant temporal contexts with a fusion mechanism that integrates retrieved memory with current edge representations.
- ETGN-RAG explicitly constructs edge embeddings by concatenating source and destination node representations, enabling direct estimation of future link probabilities through a lightweight neural predictor.
- The proposed framework incorporates time-sliced graph construction, deterministic structural node features, imbalance-aware loss optimization, and gradient clipping, ensuring scalability and robustness on real-world transactions.
- We conduct extensive experiments on large-scale cryptocurrency transaction networks and evaluate predictive performance using standard metrics such as AUC, precision, recall, and F1-score, demonstrating consistent improvements over representative static and dynamic baselines.

The remainder of this paper is organized as follows: Section II reviews related works on link prediction methods. Section III outlines the preliminaries and the problem statement. In Section IV, we present the proposed ETGN-RAG model and its components. Section V details the experimental setup and implementation specifics. Section VI provides the performance results from the conducted experiments, and finally, Section VII concludes the paper by discussing future research directions.

II. Related Works

A. Static Graph Models for Link Prediction

In static graphs, link prediction estimates the likelihood of missing or future edges using observed topology and node attributes. Traditional methods rely on structural similarity and proximity-based heuristics, leveraging neighborhood overlap and path connectivity [9]–[11]. While efficient and interpretable, these approaches perform poorly in sparse or complex networks with higher-order dependencies. To address this, graph embedding techniques learn low-dimensional node representations that preserve structural information for link inference. Representative methods include DeepWalk [5], which uses random walks, and SDNE [6], which captures nonlinear structures via deep auto-encoders. Graph Attention Networks (GAT) [7] further improve representations through adaptive neighborhood weighting. Structural anomaly detection methods such as Oddball [8] and contrastive frameworks like GCCAD [14] provide additional insights. However, all these approaches assume static snapshots and fail to capture temporal evolution and sequential dependencies, limiting their effectiveness in dynamic transaction networks.

B. Dynamic Graph Models for Link Prediction

Dynamic graph link prediction models the temporal evolution of network structures to forecast future interactions. Unlike static methods, it incorporates time-dependent information to capture sequential dependencies and evolving connectivity patterns [18]. This is critical in transaction networks, where user behavior and interaction structures change continuously [3], [4]. Representative approaches include EvolveGCN [15], which updates graph convolutional parameters over time, and Tail-GNN [16], which improves learning for sparsely connected nodes. Tensor-based methods, such as adaptive nonnegative tensor decomposition [17] model temporal dynamics through latent factorization but face scalability and nonlinear limitations. Despite these advances, existing methods often focus on short-term dynamics and node-level representations, limiting their ability to capture long-range dependencies and fine-grained edge behavior. Many rely on snapshot aggregation or parameter evolution, which weakens historical context preservation.

III. Preliminaries and Problem Statement

A. Preliminaries

A cryptocurrency transaction system is modeled as a directed temporal graph, where wallet addresses are represented as nodes and transactions are represented as directed edges. Let $G_t = (V_t, \overrightarrow{E}_t)$ denote the transaction graph corresponding to the t-th temporal slice. Each edge $e = (u, v) \in \overrightarrow{E}_t$ indicates a transaction from sender address u to receiver address v occurring within the corresponding time window.

Definition 1 (Temporal Graph Construction): Given a transaction log containing sender, receiver, and timestamp fields, transactions are grouped into fixed-width temporal slices of 7 days. For each slice t, a directed graph G_t is constructed by

inserting an edge (u, v) for every transaction occurring in that window.

Definition 2 (Node Structural Features): Each node $v \in V_t$ is represented by a deterministic structural feature vector

$$\mathbf{x}_v = [\deg_{in}(v),\ \deg_{out}(v),\ \mathrm{PR}(v),\ \mathrm{BC}(v)] \in \mathbb{R}^4, \quad (1)$$

where $\deg_{in}$ and $\deg_{out}$ denote in-degree and out-degree, $\mathrm{PR}(v)$ is PageRank centrality, and $\mathrm{BC}(v)$ is betweenness centrality computed on G_t. These features capture local connectivity, global influence, and information flow characteristics.

Definition 3 (Node and Edge Labels): Let $\alpha(v)$ denote the eigenvector centrality (affinity) of node v and $\beta(v)$ denote its out-degree centrality (anti-affinity). A node is labeled as abnormal if

$$\alpha(v) \geq \mu_\alpha \quad \vee \quad \beta(v) < \mu_\beta, \quad (2)$$

where μ_α and μ_β are the mean affinity and anti-affinity over all nodes in G_t. An edge (u, v) is labeled as abnormal if either endpoint is abnormal:

$$y_{uv} = \mathbb{I}\left(y_u = 1 \vee y_v = 1\right). \quad (3)$$

B. Problem Statement

Given a sequence of temporal graphs $\{G_t\}_{t=1}^{T}$ with node features $\mathbf{X}_t$, edge indices $\vec{E}_t$, and partially observed edge labels y_{uv}, the objective is to learn a function

$$f_\theta : (G_t, \mathbf{X}_t) \to \hat{y}_{uv}, \quad (4)$$

that predicts whether a transaction edge is *normal* or *abnormal*. If both ends of the edge are *normal* means the link exists and is given the label "0"; otherwise labeled "1", meaning not likely to exist in the future as the network evolves. The task corresponds to temporal link prediction under edge classification, where the model must: capture evolving structural dependencies across time, exploit spatial dependencies in transaction graphs, leverage historical edge embeddings using retrieval-augmented memory, and remain robust to class imbalance and sparse behaviors. The proposed Enhanced Temporal Graph Network with Retrival-Augmented Graph (ETGN-RAG) addresses these requirements by integrating graph attention, temporal recurrence, external memory retrieval, and supervised edge classification.

IV. Proposed ETGN-RAG

This section describes the architecture of the proposed ETGN-RAG model shown in Fig. 1, which consists of: (i) a graph-attention-based multi-graph LSTM encoder with missing information prediction, (ii) a retrieval-augmented graph memory module, (iii) attention-aware residual feature fusion, and (iv) an enhanced edge-level classifier.

A. Input Representation

For each temporal slice G_t, the model receives **Node feature matrix:** $\mathbf{X}_t \in \mathbb{R}^{N_t \times F}$; **Edge index matrix:** $\mathbf{E}_t \in \mathbb{N}^{2\times|\vec{E}_t|}$, and **Edge labels:** $\mathbf{y}_t \in \{0,1\}^{|\vec{E}_t|}$ (training only). Here, F denotes the number of extracted node features, including pagerank centrality, betweenness centrality, local degrees, and global graph descriptors.

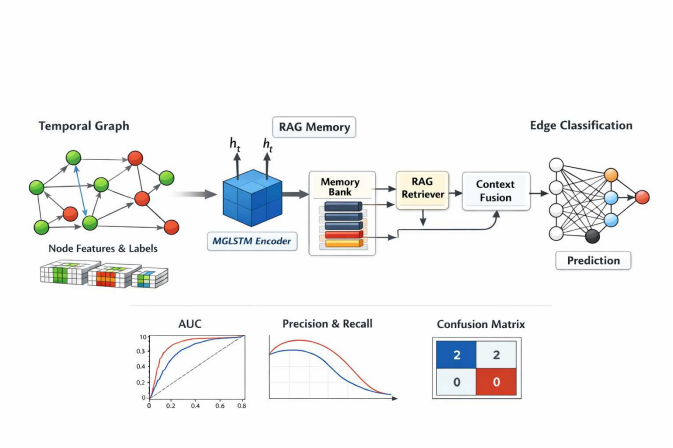


Fig. 1: Structure of the ETGN-RAG Model.

B. Multi-Scale Feature Projection

To capture heterogeneous structural patterns, node features are first projected into multiple latent spaces using parallel linear transformations:

$$\mathbf{z}_t^{(i)} = \mathbf{W}_i \mathbf{X}_t + \mathbf{b}_i, \quad i = 1, \ldots, S, \quad (5)$$

where S denotes the number of scales. The final node embedding is obtained by concatenation:

$$\mathbf{Z}_t = \mathrm{Concat}(\mathbf{z}_t^{(1)}, \ldots, \mathbf{z}_t^{(S)}). \quad (6)$$

C. Graph-Attention-Based MGLSTM with Missing Information Prediction

To jointly model spatial and temporal dependencies while correcting incomplete or noisy signals, a modified MGLSTM cell is employed. Graph attention layers replace linear transformations:

$$\mathbf{X}'_t = \mathrm{GAT}_x(\mathbf{Z}_t, \mathbf{E}_t), \quad (7)$$

$$\mathbf{H}'_{t-1} = \mathrm{GAT}_h(\mathbf{H}_{t-1}, \mathbf{E}_t). \quad (8)$$

a) Missing Information Prediction.: Due to sparse observations and delayed transactions, the propagated hidden state may contain missing or distorted information. To compensate for this, a learnable correction mechanism predicts missing information using two gating functions:

$$\boldsymbol{\gamma}_t = \tanh(\mathbf{W}_\gamma \mathbf{H}_{t-1}), \quad (9)$$

$$\boldsymbol{\beta}_t = \tanh(\mathbf{W}_\beta \mathbf{H}'_{t-1}), \quad (10)$$

and the corrected hidden signal is computed as:

$$\widehat{\mathbf{H}}_{t-1} = \mathbf{H}_{t-1} + \boldsymbol{\gamma}_t - \boldsymbol{\beta}_t. \quad (11)$$

b) LSTM Gate Update.: Using the corrected hidden representation, the LSTM gates are computed as:

$$\mathbf{f}_t = \sigma(\mathbf{X}'_t + \widehat{\mathbf{H}}_{t-1}), \quad (12)$$

$$\mathbf{i}_t = \sigma(\mathbf{X}'_t), \quad (13)$$

$$\mathbf{o}_t = \sigma(\widehat{\mathbf{H}}_{t-1}), \quad (14)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}'_t). \quad (15)$$

The memory cell and hidden state are updated as:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{16}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{17}$$

c) Attention Aggregation and Regularization.: An attention layer aggregates node embeddings across the temporal slice:

$$\alpha_i = \frac{\exp(g(\mathbf{h}_i))}{\sum_j \exp(g(\mathbf{h}_j))}, \quad \mathbf{h}_t^* = \sum_i \alpha_i \mathbf{h}_i, \tag{18}$$

where $g(\cdot)$ is a small neural network. Dropout is applied to mitigate overfitting.

D. Edge Embedding Construction

For each directed edge (u, v), the edge embedding is formed as:

$$\mathbf{e}_{uv} = \text{Concat}(\mathbf{h}_u^*, \mathbf{h}_v^*) \in \mathbb{R}^{2d}. \tag{19}$$

E. Retrieval-Augmented Graph Memory

The memory buffer stores historical edge embeddings: $\mathcal{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_M\}$. Given a query $\mathbf{q} = \mathbf{e}_{uv}$ and a memory vector $\mathbf{m}_i$, similarity is computed using cosine-normalized dot product:

$$s_i = \left\langle \frac{\mathbf{q}}{\|\mathbf{q}\|_2}, \frac{\mathbf{m}_i}{\|\mathbf{m}_i\|_2} \right\rangle. \tag{20}$$

The top-k neighbors are retrieved and averaged:

$$\mathbf{c}_{uv} = \frac{1}{k} \sum_{i \in \text{TopK}(s)} \mathbf{m}_i. \tag{21}$$

F. Residual Feature Fusion

The current edge embedding and retrieved context are fused using a residual projection:

$$\mathbf{z}_{uv} = \mathbf{W}_f[\mathbf{e}_{uv} \,\|\, \mathbf{c}_{uv}], \tag{22}$$

$$\tilde{\mathbf{e}}_{uv} = \text{ReLU}(\mathbf{z}_{uv} + \text{LayerNorm}(\mathbf{W}_r \mathbf{z}_{uv})), \tag{23}$$

which stabilizes optimization and preserves discriminative information, and Algorithm 1 summarizes the retrieval-augmented garph memory and fusion .

G. Edge Classification

The fused edge embedding is classified using a deep MLP:

$$\hat{y}_{uv} = \sigma(\text{MLP}(\tilde{\mathbf{e}}_{uv})), \tag{24}$$

where $\sigma(\cdot)$ is the sigmoid function. The model outputs the probability that edge (u, v) corresponds to a valid or normal transaction in future network evolution.

Algorithm 1: Retrieval-Augmented Graph Memory and Fusion (RAG-Fusion)

Input : Current edge embedding $\mathbf{e}_{uv} \in \mathbb{R}^{2d}$;
Memory buffer $\mathcal{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_M\}$;
Number of retrieved neighbors k;
Fusion parameters $\mathbf{W}_f, \mathbf{W}_r$;
Training flag `train`;
Optional label y_{uv}
Output: Fused edge embedding $\tilde{\mathbf{e}}_{uv} \in \mathbb{R}^{2d}$

Function RAGFusion($\mathbf{e}_{uv}, \mathcal{M}, k, \mathbf{W}_f, \mathbf{W}_r$, *train*, y_{uv}):
  $\mathbf{q} \leftarrow \mathbf{e}_{uv}$; // Query embedding
  if $|\mathcal{M}| > 0$ **then**
    foreach $\mathbf{m}_i \in \mathcal{M}$ **do**
      $s_i \leftarrow \left\langle \frac{\mathbf{q}}{\|\mathbf{q}\|_2}, \frac{\mathbf{m}_i}{\|\mathbf{m}_i\|_2} \right\rangle$; // Cosine similarity
    $\mathcal{I} \leftarrow$ TopKIndices($\{s_i\}, k$);
    $\mathcal{M}_k \leftarrow \{\mathbf{m}_i : i \in \mathcal{I}\}$;
    $\mathbf{c}_{uv} \leftarrow \frac{1}{k} \sum_{\mathbf{m}_i \in \mathcal{M}_k} \mathbf{m}_i$; // Context vector
    $\mathbf{z} \leftarrow [\mathbf{e}_{uv} \,\|\, \mathbf{c}_{uv}]$; // Concatenation
    $\mathbf{h} \leftarrow \mathbf{W}_f \mathbf{z}$; // Linear projection
    $\tilde{\mathbf{e}}_{uv} \leftarrow \text{ReLU}(\mathbf{h} + \text{LayerNorm}(\mathbf{W}_r \mathbf{h}))$; // Residual fusion
  else
    $\tilde{\mathbf{e}}_{uv} \leftarrow \mathbf{e}_{uv}$; // Cold-start case
  if *train* = *True* **then**
    $\mathcal{M}$.append($\mathbf{e}_{uv}, y_{uv}$);
    if $|\mathcal{M}| > M_{\max}$ **then**
      $\mathcal{M}$.pop_oldest();
  return $\tilde{\mathbf{e}}_{uv}$;

H. Training Objective

The model parameters are optimized using a weighted binary cross-entropy loss to address the severe class imbalance commonly observed in cryptocurrency transaction datasets. Given N training edges, the loss is defined as

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[w_+ y_i \log\left(\sigma(\hat{y}_i)\right) + (1 - y_i) \log\left(1 - \sigma(\hat{y}_i)\right) \right], \tag{25}$$

where $y_i \in \{0, 1\}$ denotes the ground-truth label of the i-th edge uv, $\hat{y}_i$ is the corresponding logit prediction, and $\sigma(\cdot)$ represents the sigmoid activation function.

The positive-class weight w_+ dynamically compensates for class imbalance and is computed at each training iteration as the ratio between the number of negative and positive samples in the current mini-batch. To stabilize optimization and prevent exploding gradients caused by recurrent propagation and graph attention operations, gradient norm clipping is applied after backpropagation:

$$\|\nabla_\theta \mathcal{L}\|_2 \leq \tau, \tag{26}$$

where θ denotes the set of trainable parameters and τ is a predefined clipping threshold.

I. Regularization and Optimization

Regularization is applied at multiple levels to improve generalization. Dropout is used in the graph attention layers of the encoder and the edge classifier to prevent neuron co-adaptation. The AdamW optimizer, which separates weight decay from updates, is utilized for stable convergence with a small weight decay coefficient to reduce overfitting.

V. Experimental Setup and Implementation

A. Transaction Network Datasets

We employed transaction networks from multiple cryptocurrency sources to analyze transaction patterns and detect the existence of future possible links. These include Ethereum-based ERC20 and ERC721 networks, as well as Bitcoin web-of-trust networks. The Ethereum networks were sourced from [1] and include ERC20 and ERC721 token transactions. These datasets cover multiple network iterations, including major stablecoins and popular ERC721 tokens, and were used by [19] and [20]. We used five blocks from 0 to 4999999 for both ERC20 & ERC721 to represent successive snapshots of the network activity over time, both from Etherscan.io.

For Bitcoin, we utilized two signed web-of-trust networks: the Bitcoin Alpha (BTA) [2] and Bitcoin (BOTC) [3] networks, both sourced from Stanford Large Network Dataset Collection. The BTA and BOTC networks capture trust ratings among users, and they were also used by [21] and [22] to provide insight into user relationships and trust-based interactions within the Bitcoin ecosystem. During implementation and analysis, each dataset was divided into 70% for training, 20% for validation, and 10% for testing to ensure robust model evaluation and reproducibility.

B. Baseline Methods

To benchmark ETGN-RAG, we compare against five advanced graph-based models spanning static and dynamic settings. EvolveGCN [15] captures temporal evolution by updating GCN parameters over time. Tail-GNN [16] improves representation learning for sparsely connected nodes. TNP [18] models temporal dependencies for future interaction prediction. VW-DBG [4] analyzes evolving Bitcoin transaction networks, while ETEA [3] examines Ethereum network dynamics to reveal interaction patterns.

C. Implementation Details

The ETGN-RAG model was implemented in Python 3.10.0 using PyTorch and PyTorch Geometric, along with libraries such as Pandas and NetworkX for data processing and visualization. Transaction data is processed into temporal graph slices, with node features extracted using in-degree, out-degree, PageRank, and betweenness centrality, and edge labels based on node affinity scores. The data is organized into 'torch_geometric.data.Data' objects and batched with 'DataLoader' for training. The architecture includes a multi-granular LSTM (MGLSTM) with GATConv layers, a retrieval-augmented graph memory (RAG), a fusion layer for embedding combinations, and an edge classifier for binary prediction. It trains using a weighted binary cross-entropy loss with gradient clipping and the AdamW optimizer.

[1] https://xblock.pro/xblock-eth.html

[2] https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html

[3] https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html

D. Evaluation Criteria

For evaluation, we calculate standard metrics including AUC, precision, recall, and F1-score, along with visualizations such as confusion matrices, ROC curves, and precision-recall curves. The source code for the ETGN-RAG implementation will be published online for reproducibility and further research.

E. Model Hyperparameters

The ETGN-RAG model uses node features of dimension 4, consisting of in-degree, out-degree, PageRank, and betweenness centrality. The hidden dimension of the MGLSTM is set to 32, with GATConv layers employing 2 attention heads and a dropout rate of 0.3. The retrieval-augmented graph memory (RAG) stores up to 5000 embeddings, with the top-5 nearest neighbors used during retrieval. Training is performed with a batch size of 8 using the AdamW optimizer with a learning rate of 1×10^{-3} and weight decay of 1×10^{-3}, while gradient clipping is applied at 2.0. The model is trained for 300 epochs.

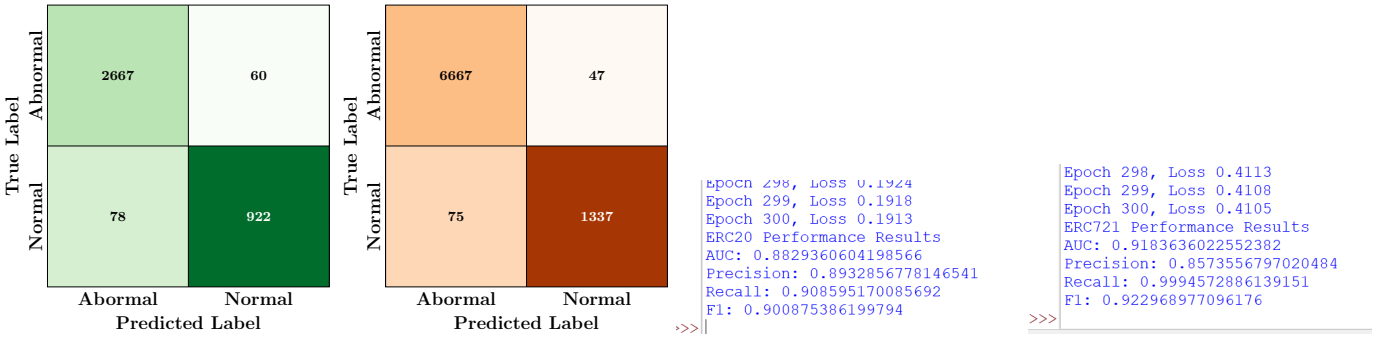
TABLE I: Performance Comparison of ETGN-RAG and Five Baseline Models.

Dataset	Criteria	EvolveGCN	Tail-GNN	TNP	VW-DBG	ETEA	ETGN-RAG
BTA	*AUC (%)*	58.9	64.7	61.3	69.5	63.8	**95.0**
	Precision (%)	62.3	60.8	65.7	68.2	59.9	**93.9**
	Recall (%)	52.4	55.9	58.6	54.1	56.3	**92.2**
	F_1 *(%)*	60.7	63.5	66.9	61.8	64.2	**92.9**
BOTC	*AUC (%)*	66.5	71.4	63.7	59.8	64.9	**97.7**
	Precision (%)	65.1	61.9	69.4	71.2	60.7	**96.6**
	Recall (%)	50.3	56.8	54.6	58.1	55.9	**94.7**
	F_1 *(%)*	58.4	67.2	62.1	69.8	65.7	**95.6**
ERC20	*AUC (%)*	60.8	57.4	70.1	74.2	55.6	**88.3**
	Precision (%)	74.2	68.3	71.5	73.9	69.1	**89.3**
	Recall (%)	56.8	58.7	63.9	61.5	59.8	**90.9**
	F_1 *(%)*	64.9	69.7	66.1	67.8	71.3	**90.1**
ERC721	*AUC (%)*	59.4	55.6	71.8	67.3	58.9	**91.8**
	Precision (%)	80.1	75.8	70.4	65.9	74.6	**85.9**
	Recall (%)	53.6	55.2	57.8	59.3	61.7	**99.9**
	F_1 *(%)*	65.8	67.9	63.7	66.1	71.9	**92.3**

VI. Experimental Results

A. Link Prediction Efficiency

The performance comparison presented in Table I clearly demonstrates the superiority of the proposed ETGN-RAG model over five state-of-the-art baseline methods like EvolveGCN, Tail-GNN, TNP, VW-DBG, and ETEA, across four blockchain cryptocurrency transaction network datasets: BTA, BOTC, ERC20, and ERC721. For the BTA dataset, ETGN-RAG achieves an AUC of 95.0%, a precision of 93.9%, a recall of 92.2%, and an F_1-score of 92.9%, significantly outperforming the best-performing baseline, VW-DBG, which attains only 69.5% AUC and 68.2% precision. Similarly, for the BOTC dataset, ETGN-RAG records an AUC of 97.7%, precision of 96.6%, recall of 94.7%, and F_1 of 95.6%, consistently exceeding all baseline results by



(a) BTA Confusion Matrix. (b) BOTC Confusion Matrix. (c) Python Screenshot for ERC20. (d) Python Screenshot for ERC721.

Fig. 2: Confusion Matrices of BTA and BOTC and Python-Shell Screenshots of the Performance Results of ERC20 and ERC721.

a substantial margin. These improvements illustrate ETGN-RAG's enhanced capability in capturing temporal dynamics and relational dependencies within transaction networks.

The trend remains consistent across the ERC20 and ERC721 datasets. ETGN-RAG achieves the highest AUC values of 88.3% and 91.8% for ERC20 and ERC721, respectively, while maintaining high precision, recall, and F_1-scores, with recall reaching an exceptional 99.9% for ERC721. In contrast, baseline models exhibit considerably lower performance, often falling below 75% in AUC and F_1 metrics. These results highlight ETGN-RAG's robustness and generalizability across diverse blockchain environments, demonstrating that the integration of retrieval-augmented graph memory and multi-graph temporal encoding significantly enhances the model's predictive accuracy. Fig. 2a and Fig. 2b present the confusion matrices of BTA and BOTC blockchain cryptocurrency transaction networks to help understand the true positives (TP), true negatives (TN), false positives (FN), and false negatives of the true labels against the predicted labels. Also, in Fig. 2c and Fig. 2d, we presented the Python-shell screenshots of the performance results of ETGN-RAG on ERC20 and ERC721 cryptocurrency transaction network datasets based on AUC, precision, recall, and f1-score.

TABLE II: Hyperparameter Analysis of ETGN-RAG Model.

Dataset	Hyperparameter	Value	AUC (%)
BTA	Batch Size	8	88.9
		16	90.7
		32	**95.0**
		64	92.1
BOTC	Learning Rate	0.1	92.3
		0.01	93.1
		0.001	**97.7**
		0.0001	91.6
ERC20	Epochs	100	80.7
		200	83.4
		300	**88.3**
		400	85.5
ERC721	GAT Attention Heads	1	77.8
		2	**91.8**
		3	86.2
		4	87.3

B. Hyperparameter Sensitivity

Table II presents a comprehensive hyperparameter sensitivity analysis of the proposed ETGN-RAG model across four benchmark datasets, where the optimal configurations are highlighted in **bold**. For the BTA dataset, a batch size of **32** achieves the highest AUC of **95.0%**, indicating an effective balance between gradient stability and generalization performance. On the BOTC dataset, the learning rate of **0.001** yields the best AUC of **97.7%**, confirming that moderate step sizes enable more stable convergence and superior discrimination capability. For the ERC20 dataset, training the model for **300** epochs maximizes performance with an AUC of **88.3%**, suggesting sufficient model convergence without overfitting. Similarly, on the ERC721 dataset, using **2** GAT attention heads produces the highest AUC of **91.8%**, demonstrating that moderate attention capacity effectively captures relational dependencies while avoiding excessive complexity.

TABLE III: Analysis of Ablation Experiments.

Dataset	Ablation	AUC%
BTA	**ETGN-RAG**	**95.0**
	Without GAT layers	84.3
	Without LSTM gates	79.9
	Without RAG memory	86.2
	Without Missing information prediction	92.5
BOTC	**ETGN-RAG**	**97.7**
	Without GAT layers	89.6
	Without LSTM gates	85.9
	Without RAG memory	90.8
	Without Missing information prediction	93.3

C. Ablation Experiments

To assess the contribution of each module in ETGN-RAG, we conduct ablation experiments by removing the GAT layers, LSTM gates, RAG memory, and missing information prediction module. As shown in Table III, the full ETGN-RAG model achieves the highest AUC scores of 95.0% on BTA and 97.7% on BOTC, validating the effectiveness of the integrated architecture. Removing the GAT layers causes a notable performance drop, highlighting the importance of

graph attention for structural representation learning. Eliminating the LSTM gates results in the most significant degradation, demonstrating the critical role of temporal gating in modeling sequential dependencies. Excluding the RAG memory module also substantially reduces performance, confirming the benefit of retrieval-augmented contextual reasoning. In contrast, removing the missing information prediction module yields only a moderate decline, indicating a complementary contribution.

TABLE IV: Effect of Perturbation Attack on ETGN-RAG Performance (AUC %).

Perturbation Level	ETGN-RAG	5%	10%	15%
BTA	**95.0**	94.6	93.8	91.5
BOTC	**97.7**	97.2	96.8	96.3
ERC20	**88.3**	87.9	87.4	87.0
ERC721	**91.8**	89.9	88.6	88.2

D. Robustness Testing

We tested the robustness of the ETGN-RAG model through a perturbation attack by introducing controlled modifications to the graph structure and edge attributes. Applying perturbation levels of 5%, 10%, and 15% to randomly selected edges, we measured performance while maintaining overall graph plausibility. As shown in Table IV, increasing perturbation levels progressively degrade ETGN-RAG's performance, yet even at 15% perturbation, the model retains high AUC values, demonstrating its resilience to adversarial modifications in dynamic graph structures.

VII. Conclusion

In this work, we proposed ETGN-RAG, an Enhanced Temporal Graph Network augmented with a retrieval-based graph memory, for accurate edge-level link prediction in dynamic transaction networks. By integrating multi-graph LSTM modules with GAT-based attention, retrieval-augmented graph memory, and a missing information prediction module, ETGN-RAG effectively captures both temporal dependencies and long-range structural patterns. Our experimental results on BTA, BOTC, ERC20, and ERC721 datasets demonstrate superior performance in terms of AUC, Precision, Recall, and F1-score, outperforming several state-of-the-art baseline models. Ablation and perturbation analyses further highlight the critical contributions of the GAT layers, LSTM gates, and memory modules, as well as the model's robustness to adversarial modifications. Future work includes exploring multi-scale temporal modeling to capture fine-grained and coarse-grained patterns and also incorporating adaptive attention mechanisms simultaneously, which could further enhance prediction accuracy in highly dynamic networks.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf. Accessed: Sept. 26, 2025.

[2] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Yellow Paper, Shanghai Version 47e97f5, pp. 1–42, 2024. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf. Accessed: Sept. 26, 2025.

[3] D. Lin, J. Chen, J. Wu, and Z. Zheng, "Evolution of Ethereum transaction relationships: Toward understanding global driving factors from microscopic patterns," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 2, pp. 559–570, 2022.

[4] J. Geng, Y. Li, L. Fang, and P. Chen, "VW-DBG: A dynamically evolving Bitcoin transaction network model," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 2, pp. 356–363, 2022, doi: 10.1109/TNSE.2021.3116243.

[5] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, pp. 701–710.

[6] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2016, pp. 1225–1234.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[8] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Advances in Knowledge Discovery and Data Mining*, LNCS, vol. 6119. Berlin, Germany: Springer, 2010, pp. 410–421.

[9] S. Huang and L. Ma, "Social network link prediction algorithm based on node similarity," in *Proc. IEEE Asia-Pacific Conf. Image Processing, Electronics and Computers (IPEC)*, Dalian, China, 2022, pp. 1357–1360.

[10] T. Li, C. Zeng, Y. Feng, Y. Zhang, and K. Wang, "Research of local similarity index based on OWA integration operator in terrorist network link prediction method," in *Proc. 33rd Chinese Control and Decision Conf. (CCDC)*, Kunming, China, 2021, pp. 6420–6424.

[11] W. Chen and Y. Zhou, "A link prediction similarity index based on enhanced local path method," in *Proc. 40th Chinese Control Conf. (CCC)*, Shanghai, China, 2021, pp. 753–757.

[12] A. Jibouni, D. Lotfi, and A. D. E. Maliani, "Link prediction in weighted complex networks using machine learning methods and similarity metrics," in *Proc. 10th Int. Conf. Wireless Networks and Mobile Communications (WINCOM)*, Istanbul, Türkiye, 2023, pp. 1–6.

[13] P. Ashok, G. N. Jagadeesha, and V. P. Baligar, "Link prediction in social networks using proximity-based algorithms," in *Proc. 4th Int. Conf. Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, Mandya, India, 2022, pp. 1–6.

[14] B. Chen, J. Zhang, X. Zhang, Y. Dong, J. Song, P. Zhang, K. Xu, E. Kharlamov, and J. Tang, GCCAD: Graph Contrastive Coding for Anomaly Detection, *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 8037-8051, no. 8, AUGUST 2023.

[15] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 34, no. 4, 2020, pp. 5363–5370.

[16] Z. Liu, T.-K. Nguyen, and Y. Fang, "Tail-GNN: Tail-node graph neural networks," in *Proc. 27th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, New York, NY, USA, 2021, pp. 1109–1119.

[17] H. Wu and W. Li, "Link prediction for dynamic weighted graph via adaptive nonnegative tensor CP decomposition," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, Kuching, Malaysia, 2024, pp. 3462–3466.

[18] L. Zou, X.-X. Zhan, J. Sun, A. Hanjalic, and H. Wang, "Temporal network prediction and interpretation," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1215–1224, 2022.

[19] A. Badjie, G. M. Ntuala, Q. Xia, J. Gao, and H. Xia, "MGGPT: A Multi-Graph GPT-Enhanced Framework for Dynamic Fraud Detection in Cryptocurrency Networks," *Computer Networks*, vol. 270, p. 111508, 2025, doi: 10.1016/j.comnet.2025.111508.

[20] J. Gao, A. Badjie, Q. Xia, P. Mukala, H. Xia, and G. M. Ntuala, "Advanced Temporal Graph Embedding for Detecting Fraudulent Transactions on Complex Blockchain Transactional Networks," in *Information Security and Privacy, Lecture Notes in Computer Science*, vol. 15658, 2025, pp. 280–297, doi: 10.1007/978-981-96-9095-4_17.

[21] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. S. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *Proc. 11th ACM Int. Conf. on Web Search and Data Mining (WSDM)*, 2018, pp. 333–341.

[22] J. Gao, A. Badjie, Q. Xia, M. N. Grace, and H. Xia, "Optimized Temporal Graph Embedding for Detecting Fraudulent Transactions," *Tsinghua Science and Technology*, 2025, doi: 10.26599/TST.2025.9010141.