International Journal of Communication and Information Technology

E-ISSN: 2707-6628
P-ISSN: 2707-661X
www.computersciencejournals.com/ijcit
IJCIT 2023; 4(2): 33-39
Received: 07-06-2023
Accepted: 11-07-2023

Akhila Reddy Yadulla
Information Technology, University of the Cumberlands, Williamsburg, KY, USA

Mounica Yenugula
Information Technology, University of the Cumberlands, Williamsburg, KY, USA

Vinay Kumar Kasula
Information Technology, University of the Cumberlands, Williamsburg, KY, USA

Bhargavi Konda
Information Technology, University of the Cumberlands, Williamsburg, KY, USA

Santosh Reddy Addula
Information Technology, University of the Cumberlands, Williamsburg, KY, USA

Sarath Babu Rakki
JNTUH College of Engineering, Science and Technology, Hyderabad, Telangana, India

Corresponding Author:
Akhila Reddy Yadulla
Information Technology, University of the Cumberlands, Williamsburg, KY, USA

A time-aware LSTM model for detecting criminal activities in blockchain transactions

Akhila Reddy Yadulla, Mounica Yenugula, Vinay Kumar Kasula, Bhargavi Konda, Santosh Reddy Addula and Sarath Babu Rakki

DOI: https://doi.org/10.33545/2707661X.2023.v4.i2a.108

Abstract
This paper introduces the Time-Aware LSTM (T-LSTM) model to identify criminal activities involving USDT on wallet addresses within the blockchain ecosystem. The model utilizes a time-aware LSTM architecture to learn the continuous variations in node address features over different transaction time intervals. Additionally, a gating mechanism filters the influence intensity of neighboring transaction node addresses on the central node. The gating mechanism accounts for the transactional correlation strength between node addresses. Finally, a self-attention mechanism is employed to integrate node address features across various transaction timestamps, producing a comprehensive feature representation for the addresses. Experimental results demonstrate that the T-LSTM model effectively captures the dynamic feature changes of node addresses over irregular transaction intervals, outperforming traditional detection models regarding precision, recall, and F1 score on the test set.

Keywords: Time-Aware LSTM, temporal model, gating mechanism, self-attention mechanism

Introduction
Blockchain technology, recognized as one of the most transformative innovations of the 21st century, is reshaping global financial and technological ecosystems. Its decentralized and anonymous nature has brought numerous benefits but also posed significant challenges, particularly with the rise of cybercrime. A major concern is the use of Tether (USDT) for illegal activities, as its widespread adoption as a global payment method has led to increasing misuse. Identifying and addressing criminal activities involving USDT on blockchain networks has thus become a pressing challenge. This study focuses on USDT transactions on the TRON blockchain and proposes a novel detection method for identifying criminal behavior. TRON (TRX) is a blockchain-based cryptocurrency that utilizes the Delegated Proof-of-Stake (DPoS) consensus algorithm to enable high-speed transactions and scalability. As a result, TRON has become a popular platform for USDT transactions. The transactional structure of TRON aligns well with graph-based representations, where wallet addresses serve as nodes and transactions form edges. This structure makes graph-based models, particularly Graph Neural Networks (GNNs), an effective choice for analyzing blockchain data. GNNs are capable of capturing the complex relationships between nodes in a graph, making them highly suitable for tasks like blockchain transaction analysis.

Existing research has applied GNNs to blockchain analysis in areas such as anomaly detection, account classification, and transaction tracing [1]. For example, Xia *et al.* [2] proposed an adaptive graph embedding algorithm for detecting malicious accounts in Ethereum by constructing transaction network structures and attribute features. Chen *et al.* [3] used Gated Recurrent Unit (GRU) networks to capture temporal features in transaction sequences, enabling the identification of phishing activities in Ethereum. Choi *et al.* [4] developed the DeepWalk-Transformer (DGTSD) model, which leverages Transformer's multi-head attention mechanism to analyze complex patterns in Ethereum transaction graphs for fraud detection. Similarly, Liu *et al.* [5] introduced the GTN2vec model, combining biased random walks and transaction network structures to extract risk-related node features. Kim *et al.* [6] proposed the ATGraph model, a heterogeneous graph representation of blockchain transactions, to identify high-risk accounts by learning multi-relational node representations.

Despite the success of these methods, they often overlook the temporal dynamics of blockchain transactions. The continuous evolution of node features over time, as well as the

varying influence of neighboring nodes on a central node during different transaction intervals, remains underexplored.

To address these limitations, this study introduces the Time-Aware LSTM (T-LSTM) model, a temporal sequence model designed to capture the dynamic changes in node address features over irregular transaction intervals. By learning the continuous evolution of features and incorporating temporal dependencies, the T-LSTM model can effectively detect criminal activities in USDT transactions on the TRON blockchain. The model improves upon existing approaches by accounting for the temporal relationships between transaction events, offering better accuracy and efficiency in identifying criminal addresses.

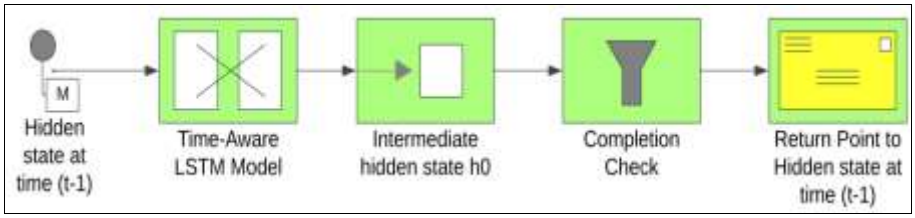

Fig 1: T-LSTM Model Workflow

Related Background Knowledge

Differential equations are widely used to describe complex continuous processes, modeling the changing patterns of variables within a given process, as shown in Equation (1).

$$\frac{dy}{dt} = f_\theta(y(t), t) \quad (1)$$

Here, $y(t)$ is the unknown function, t is the independent variable, and f_θ describes the rate of change of the derivative at $y(t)$ at time t. The Time-Aware LSTM (T-LSTM) model is a deep learning approach that incorporates temporal dependencies to capture the evolving patterns of data over time. While traditional neural networks such as LSTMs are designed to learn temporal sequences, the T-LSTM model adapts to irregular time intervals, making it suitable for dynamic and time-sensitive tasks.

The Time-Aware LSTM model improves upon conventional LSTM networks by explicitly integrating the time intervals between events into the learning process [7]. This allows the model to better handle data with varying temporal gaps. The temporal evolution of node features in a blockchain transaction context can be modeled using the following recurrent equation:

$$h_t = LSTM(h_{t-1}, x_t, \Delta t_t) \quad (2)$$

Here, h_t represents the hidden state of the LSTM at the time t, x_t is the input feature vector at the time t, and Δt_t is the time interval between consecutive transactions or events. The inclusion of Δt_t allows the model to account for the temporal gaps between data points, providing a more accurate representation of the time-evolving data.

The final output of the T-LSTM model can be expressed as:

$$y(T) = f_\theta(h_T, t) \quad (3)$$

Where $y(T)$ is the predicted output at the time T, h_T is the final hidden state of the LSTM, and f_θ is the learned function that maps the temporal features to the desired prediction (e.g., criminal activity detection).

Through this approach, the T-LSTM model can effectively capture the continuous evolution of features over time, making it well-suited for tasks such as predicting and detecting anomalies or criminal activities in time-sensitive domains like blockchain transactions.

Model Structure Design Principle

The Time-Aware LSTM (T-LSTM) model proposed in this paper is used to identify malicious addresses associated with risky behaviors in transactions on the TRON blockchain. The transaction evolution process of node addresses is modeled using an LSTM-based temporal model, which dynamically learns the features that change over transaction time [8]. By leveraging graph-based methods to aggregate node features, the model introduces the strength of relationships between transaction node addresses and a gating mechanism along the transaction timeline. This enables the fusion of features from neighboring node addresses. The model is designed to better capture the impact of historical information and evolutionary mechanisms on node representations. In this paper, the model is referred to as T-LSTM.

The workflow of the T-LSTM model is shown in Figure 1. Initially, the hidden state at the time $t-1$ is represented as h_{t-1}. Using the Time-Aware LSTM model, the intermediate hidden state $\hat{h}_0$ is obtained, as shown in the following equation:

$$h_0 = LSTM(h_{t-1}, x_0, \Delta t_0) \quad (4)$$

Here, h_0 is the hidden state at the time $t = 0$, x_0 is the feature vector of the central node address, and Δt_0 is the time interval between transactions. The gating mechanism is applied to fuse the features of the central node address x_0 with the hidden state, resulting in the final hidden state h_0.

Next, h_0 is used as input to the LSTM model at the next time step, producing the intermediate hidden state $\hat{h}_1$, as shown in the following equation:

$$h_1 = LSTM(h_0, x_1, \Delta t_1) \quad (5)$$

In this equation, h_1 is the hidden state at the time $t = 1$, x_1 is the feature vector of the node address x_1, which interacts with the central node x_0 at time $t = 1$, and Δt_1 is the time interval between the transactions involving these nodes.

This process continues iteratively, with the hidden state evolving over each time step until the final hidden state h_T is obtained at the time T:

$$h_T = LSTM(h_{T-1}, x_T, \Delta t_T) \quad (6)$$

Finally, the hidden states h_t at different time points are fused using a self-attention mechanism to generate the feature representation of the node address x_0. This final feature representation is used for downstream tasks such as criminal activity detection:s

$$\hat{x_0} = SelfAttention(h_0, h_1, \ldots, h_T) \quad (7)$$

Through this process, the T-LSTM model captures the continuous evolution of features over time [9, 10], enabling more accurate detection of patterns and behaviors in blockchain transactions.

T-LSTM Model Composition and Working Principle

The Time-Aware LSTM (T-LSTM) model is built upon a dynamic temporal sequence environment, where the node addresses change over time due to transaction dynamics. The core idea is to identify the central node's address and track how its features evolve with time. The model is primarily divided into three parts:

- It establishes a temporal model based on LSTM (Long Short-Term Memory).
- It constructs a gating mechanism using transaction relationships between node addresses to determine the degree of feature retention for neighboring nodes at different time steps.
- It integrates the self-attention mechanism to merge node states across time steps, ultimately generating feature representations for the central node's address.

LSTM Module

To address the irregular transaction intervals between node addresses in the wave field, conventional temporal modeling techniques (Like traditional neural networks) struggle to capture this aspect effectively. Therefore, the T-LSTM model utilizes LSTM for modeling node address changes over irregular transaction intervals [11]. The LSTM-based differential equation model evolves the node features with time and adapts using learned parameters. The equation for this is as follows:

$$\frac{dy}{dt} = LSTM(y(t), t) \quad (8)$$

Where the LSTM function replaces the conventional hidden state evolution function to account for non-uniform time intervals between transactions.

LSTM Model Principle

The LSTM is a time-sequential model consisting of several gating mechanisms, as shown by the following equations:

$$z_t = \sigma(W_z\, h_{t-1} + b_z) \quad (9)$$

$$r_t = \sigma(W_r\, h_{t-1} + b_r) \quad (10)$$

$$\hat{h}_t = tanh(W_h\, (r_t \odot h_{t-1}) + b_h) \quad (11)$$

$$h_t = z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1} \quad (12)$$

Where z_t is the output update gate vector, r_t is the reset gate vector, $\hat{h}_t$ is the candidate's hidden state, h_t is the final output hidden state.

The update gate controls the retention of information from the previous time step, and the reset gate determines how much of the previous hidden state is discarded. The memory retention involves two steps, using the same update gate z_t for filtering the candidate features $\hat{h_t}$ and previous state h_{t-1}, ultimately outputting the current state h_t.

LSTM Model Operation

In the T-LSTM model, the LSTM module generates the hidden state $\hat{h}_t$ for node addresses evolving over transaction times. The differential equation function is set to LSTM with trainable parameters. The Euler method is employed for numerical approximation to calculate the intermediate hidden state at the time t from the previous time step $t-1$, as given by the equation:

$$\hat{h}_t = \mathrm{ODESolve}(\mathrm{LSTM}, h_{t-1}, [t, t-1]) \quad (13)$$

Here, $\hat{h}_t$ is not the initial value of the LSTM differential equation at time t. Instead, the model assumes different initial value curves for each time interval, and the gating mechanism further activates the hidden state at time t, providing a refined depiction of feature evolution.

Gating Mechanism for Transaction Relationships between Node Addresses

To capture the varying influence of neighboring nodes' addresses on the central node's feature representation at different transaction times, the T-LSTM model utilizes a dynamic gating mechanism. This mechanism controls the extent to which historical information is retained and new information is integrated. The strength of the relationship between node addresses plays a pivotal role in building this gating mechanism.

The model targets time-series node address transactions, where neighboring node addresses influence the central node's address. To measure the strength of this influence, the following methodology is proposed to compute the relationship strength:

The relationship strength consists of two factors:

- The historical transaction strength between node addresses.
- The correlation between the central node's address and neighboring node addresses.

Influence of Neighboring Node Addresses

The influence of neighboring node addresses is calculated as follows:

$$\tau_{ij} = \frac{N_{ij}}{\Delta t} + \log\left(\sum \frac{V_{ij}}{N_{ij}}\right) + \log V_{ij} \quad (14)$$

where N_{ij} represents the historical transaction count between node addresses i and j, Δt is the time interval between the first transaction of the node i and node j and the current transaction time, $\sum V_{ij}$ is the total historical transaction amount between node i and node j, while V_{ij} represents the transaction amount at time t.

Thus, τ_{ij} quantifies the influence of node j on node i.

Correlation between Node Addresses

The correlation between two node addresses is computed using the common trading partners, as given by:

$$s_{ij} = \frac{\Gamma(i) \cap \Gamma(j)}{\Gamma(i) \cup \Gamma(j)} \quad (15)$$

where $\Gamma(i)$ represents the set of neighboring addresses with which the node i has transacted before the time t, $\Gamma(j)$ represents the set of neighboring addresses with which the node j has transacted before the time t.

This intersection and union measure gives a correlation coefficient between node addresses i and j, which ranges from 0 to 1 based on the overlap of trading partners.

The transaction relationship strength between node addresses is computed using the influence and correlation metrics:

$$u = \frac{1}{1 + exp(-W[\tau_{ij}, s_{ij}])} \quad (16)$$

where W is a learnable parameter, and u is the final normalized output representing the strength of the dynamic relationship.

Gating Mechanism Implementation

For a central node address i, multiple neighboring nodes j may have transactions at different time points. To ensure the influence of neighboring node j is appropriately captured in feature selection, a gating mechanism is implemented as follows:

$$h_t = u \odot W_{in} x_t + (1 - u) \odot \hat{h}_t \quad (17)$$

Here, u is the gating parameter calculated from the relationship strength. The u factor determines the weight of the features carried by the node j, influencing the feature fusion over time.

Temporal Node Address Feature Fusion with Attention Mechanism: At each time step t, the T-LSTM model outputs the hidden state h_t. To enhance the expression of node address features, a self-attention mechanism is employed. The feature sequence m_t from each time step is processed as follows:

$$Q = W_Q M, K = W_K M, V = W_V M \quad (18)$$

The self-attention matrix is then computed, and the coefficients are normalized using Softmax:

$$o = mean\left(Softmax\left(Q\frac{K^T}{d}\right)V\right) \quad (19)$$

Where Q, K, V are the query, key, and value matrices, respectively, d is the dimension of the attention scores.

The final output o represents the fused feature vector of the central node's address, derived from the weighted aggregation of its neighboring addresses across time.

Loss Function Construction

The loss calculation for the T-LSTM model is defined as:

$$\hat{y}_i = \sigma(W_{out} o_i + b_{out}) \quad (20)$$

Where $\hat{y}_i$ is the predicted output probability for the node address i. The loss is computed using cross-entropy classification:

$$Loss = -\frac{1}{N}\sum_{i \in N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \quad (21)$$

Where y_i is the label for the sample i, and the model is trained by optimizing this loss.

Algorithm 1: Dynamic Temporal Node Representation using LSTM

Input: $G = \{(x_0, x_1 \dots T)\}$, where x_0 is the initial feature of the central node's address, and $x_1 \dots T$ are the features of neighboring node addresses at each time step.

Output: $x_0 \rightarrow o$, the transformed feature representation of the central node's address.

1. Initialize $h_{-1} = 0$
2. For $t \in 0, 1, 2, \dots, T$:

- $\hat{h}_t = ODESolve(LSTM, h_{t-1}, [t, t-1])$
- $u = \frac{1}{1 + exp(-W[\tau_{ij}, s_{ij}])}$
- $h_t = u \odot W_{in} x_t + (1 - u) \odot \hat{h}_t$
- $m_t = W_o h_t + b_o$

1. End for
2. $o = Attention(m_1 \dots T)$
3. Return o (final feature representation).

Dataset Preparation and Feature Processing

Dataset Collection

The dataset was collected through the process of handling various cryptocurrency-related cases, accumulating 2,450 malicious addresses linked to risk behaviors in the TRON blockchain. These addresses primarily come from data related to telecom fraud, online gambling, and illegal currency exchanges. The 2,450 addresses were treated as labeled data, with the central node address identified as the target.

To extract relevant features from criminal transactions, first-order neighbor sampling was performed based on the transaction time of the node address. The sampling rule follows a temporal progression, capturing neighboring addresses involved in USDT transactions under the TRC-20 protocol. After eliminating duplicates, a total of 250,390 addresses were found to have USDT transactions with the central node address within the relevant crime time range. Subsequently, 2,450 normal addresses were randomly selected as labeled data. Applying the same sampling method, 219,780 additional node addresses were obtained.

As a result, the final training dataset consists of:

- 2,450 risk-labeled addresses,
- 250,390 first-order neighboring addresses,
- 2,450 normal transaction addresses,
- 219,780 first-order neighboring addresses.

Feature Processing of the Dataset: The initial features of node addresses play a crucial role in determining the

effectiveness of model training and prediction accuracy. It is essential to scientifically establish a set of initial features that can reflect the core properties of node address transactions. To accurately capture the transaction behavior of node addresses, the feature construction process integrates graph networks and statistical indicators, while also considering the influence of TRON coins (TRX) and Tether (USDT) in maintaining the proper operation of the TRON blockchain. Consequently, the feature construction process accounts for the relationship between TRON coins and USDT transactions. As a result, each node address is assigned 30-dimensional features as initial attributes. The first 25 dimensions are created using TRON's native currency (TRX) and Tether (USDT) using the same calculation method. To simplify the explanation, the focus is on feature construction for USDT, with TRX indicated in parentheses where applicable. The specific feature construction methods are as follows:

- Outdegree of Node Address for USDT (TRX): The number of transactions where the node address sends USDT (TRX), denoted as $USDT\ (TRX)_{Out_{Degree}}$.
- Indegree of Node Address for USDT (TRX): The number of transactions where the node address receives USDT (TRX), denoted as $USDT\ (TRX)_{In_{Degree}}$.
- Total Outbound USDT (TRX) Volume: The total amount of USDT (TRX) sent by the node address, denoted as $USDT\ (TRX)_{Out_{Value}}$.
- Total Inbound USDT (TRX) Volume: The total amount of USDT (TRX) received by the node address, denoted as $USDT\ (TRX)_{In_{Value}}$.
- Average Outbound USDT (TRX) Volume: The average amount of USDT (TRX) sent by the node address, calculated as $USDT(TRX)_{Out_{Value}}/USDT(TRX)_{Out_{Degree}}$, denoted as $USDT_{AvgOutValue}$.
- Average Inbound USDT (TRX) Volume: The average amount of USDT (TRX) received by the node address, calculated as $USDT\ (TRX)_{In_{Value}}\ /\ USDT\ (TRX)_{In_{Degree}}$, denoted as $AvgIn_{Value}$.
- Minimum Outbound USDT (TRX) Volume: The minimum outbound transaction value of USDT (TRX) in the historical transactions, denoted as $USDT\ (TRX)_{Min_{OutValue}}$.
- Maximum Outbound USDT (TRX) Volume: The maximum outbound transaction value of USDT (TRX) in the historical transactions, denoted as $USDT\ (TRX)_{Max_{OutValue}}$.
- Minimum Inbound USDT (TRX) Volume: The minimum inbound transaction value of USDT (TRX) in the historical transactions, denoted as $USDT\ (TRX)_{Min_{InValue}}$.
- Maximum Inbound USDT (TRX) Volume: The maximum inbound transaction value of USDT (TRX) in the historical transactions, denoted as $USDT\ (TRX)_{Max_{InValue}}$.
- Ratio of Outdegree for USDT to TRX: The ratio of $USDT_{Out_{Degree}}$ to $TRX_{Out_{Degree}}$.
- Ratio of Indegree for USDT to TRX: The ratio of $USDT_{In_{Degree}}$ to $TRX_{In_{Degree}}$.
- Ratio of Total Outbound USDT Volume to TRX Volume: The ratio of $USDT_{Out_{Value}}$ to $TRX_{Out_{Value}}$.
- Ratio of Total Inbound USDT Volume to TRX Volume: The ratio of $USDT_{In_{Value}}$ to $TRX_{In_{Value}}$.
- Ratio of Average Outbound USDT Volume to TRX Volume: The ratio of $USDT_{AvgOutValue}$ to $TRX_{AvgOutValue}$.
- Ratio of Average Inbound USDT Volume to TRX Volume: The ratio of $USDT_{AvgInValue}$ to $TRX_{AvgInValue}$.

Experimental Results and Analysis

Experimental Environment

The experiment was conducted on a Windows 10 64-bit operating system, using a CPU Intel i7-11700K and a GPU GeForce RTX 3070. The Time-Aware LSTM (T-LSTM) model code was implemented using Python, utilizing the PyTorch deep learning framework along with the DGL graph neural network library. The development environment used was Visual Studio Code.

Evaluation Metrics

In this experiment, the performance of the model was evaluated using three common metrics: Precision, Recall, and F1 score, as defined in equations (22) to (24):

Precision

$$Precision = \frac{TP}{TP+FP} \tag{22}$$

Recall

$$Recall = \frac{TP}{TP+FN} \tag{23}$$

F1 Score

$$\mathrm{F1} = \frac{2\times \mathrm{Precision}\times \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} \tag{24}$$

Where TP represents the number of true positive samples predicted correctly, FP represents the number of false positive samples predicted incorrectly, FN represents the number of false negative samples predicted incorrectly.

Table 1: Model Comparison Experiment Results

Model	Precision	Recall	F1 Score
GCN	80.25%	72.40%	75.10%
GAT	83.12%	77.50%	80.56%
T-LSTM	85.43%	80.25%	82.85%

Table 1 demonstrates the performance of the T-LSTM model in detecting malicious behaviors linked to risk in TRON blockchain node addresses. To benchmark the model's performance, traditional graph neural network models such as GCN and GAT were employed as baseline models. Precision, Recall, and F1 scores were used as evaluation metrics for the test set. Upon analyzing these metrics, it was found that the T-LSTM model outperformed both GCN and GAT across all three evaluation metrics. This highlights the capability of the T-LSTM model to capture

dynamic changes in node features and interactions over time, significantly improving detection performance.

Experimental Parameter Settings

In the experiment, the following parameters were set:

- **Batch size:** 16 samples per batch.
- **Number of iterations:** 300.
- **Optimizer:** Adam optimizer.
- **Learning rate:** 0.0005.

These settings were used to train the T-LSTM model effectively within the given environment, ensuring optimal performance.

Experimental Training Process and Effectiveness Evaluation: The progression of the loss function on the training set and the precision on the test set as the number of iterations increases during the T-LSTM model's training process. The training was conducted for 300 iterations, and the loss value dropped sharply in the first 60 steps while precision increased rapidly. After around 150 iterations, both the loss curve and precision curve began to stabilize, indicating that the model had learned well. No significant changes were observed after 300 iterations, suggesting that the model had converged.

Ablation Experiment Analysis: Ablation experiments were performed to assess the impact of each component in the T-LSTM model on its recognition ability. The following configurations were tested:

- **T-LSTM-Seq:** Replacing the neural differential equation module with a standard sequential model, such as LSTM, for temporal modeling.
- **T-LSTM-NoGate:** Disabling the gating mechanism, which was replaced with a simpler form of feature fusion: $W_{in}\ x_t + \hat{h}_t$.
- **T-LSTM-GC:** Using only the correlations between node addresses without considering neighbor features.
- **T-LSTM-NC:** Using only the influence of neighboring node addresses.

Table 2 shows the ablation experiment results, illustrating the contribution of each component to the final model performance. The advantage of the neural differential equation module in modeling irregular transaction times is evident, as T-LSTM-NoGate showed a 5.21% improvement in F1 score over T-LSTM-Seq, proving that the neural differential equation effectively captures node feature evolution. Additionally, the introduction of the gating mechanism improved feature extraction capabilities. T-LSTM-GC and T-LSTM-NC showed a 2.45% and 3.05% improvement in F1 score, respectively, compared to T-LSTM-NoGate, further highlighting the gating mechanism's ability to refine the selection of relevant node and neighbor features for the task.

Table 2: Ablation Comparison Experiment Results

Model	Precision	Recall	F1 Score
T-GRU	71.47%	64.78%	67.96%
T-ODE	76.25%	72.43%	74.29%
Tgs-ODE	79.15%	75.43%	77.25%
Tgn-ODE	78.32%	76.02%	77.15%
T-LSTM	82.76%	77.65%	80.12%

Node Feature Encoding Visualization

To better understand the model's performance, the T-LSTM model's node feature encoding was visualized using t-SNE. This technique maps the high-dimensional feature vectors of node addresses to a two-dimensional space. A random sample of 500 test samples was chosen, consisting of 250 positive and 250 negative samples. The original node feature encoding, the encoding produced by the GCN model, and the encoding produced by the GAT model, respectively. The encoding produced by the T-LSTM model. The T-LSTM model was able to more effectively cluster similar node features, and the separation between risky and normal addresses was more distinct compared to the other models.

Risk Behavior Pattern Analysis

To further explore the practical value of the T-LSTM model in detecting risk behaviors, an analysis was conducted on the node addresses identified as exhibiting risky behavior. Four distinct risk behavior patterns were identified:

- **Circular Lending Pattern:** Risky node addresses alternate transactions with fixed addresses. 27.5% of incoming transactions are from withdrawals by exchange users, and 33.8% of outgoing transactions go to exchange users. This behavior represents criminal groups' hoarding of coins, which accounts for approximately 14.2% of the test set.
- **Wandering Pattern:** Risky node addresses engage in high-frequency, small-value transactions for a period of time and then send the funds to a fixed address in a consistent cycle. This pattern is often seen in gambling websites and accounts for 9.7% of the test set.
- **Speculation Pattern:** Risky node addresses have no repeated transaction counterparts, but the correlation between the incoming and outgoing node addresses is almost always greater than 0.7. This pattern is commonly associated with illicit financial activities involving cross-border networks and represents around 7.4% of the test set.
- **Zeroing Pattern:** Risky node addresses engage in single-entry, single-exit transactions with little to no balance left. Transaction intervals are usually in the range of minutes. These addresses are often used as intermediary points for criminal activities, representing 6.5% of the test set.

Although the sample size in this section may not have sufficient statistical significance, it further validates the ability of the T-LSTM model to identify various risk behavior patterns.

Conclusion: In this study, the T-LSTM model was introduced, which integrates a trainable neural differential

equation module to fit the evolving node feature representation of TRON addresses as transaction intervals become irregular. The model also incorporates a gating mechanism based on transaction account correlation strength, allowing it to more effectively integrate features from neighboring node addresses along the transaction timeline. Experimental results confirm that the T-LSTM model captures the dynamic evolution of node features over time, greatly enhancing its ability to detect risky behaviors in the TRON blockchain through USDT transactions.

References

1. Azmeera R, Tumma C, *et al*. Enhancing blockchain communication with named data networking: A novel node model and information transmission mechanism. J Recent Trends Comput Sci Eng. 2022;10(1):35-53.
2. Zhang X, Wang Y, Li L, *et al*. A survey on virtual currency crime detection and prevention. J Cybersecurity Privacy. 2023;2(1):10-22.
3. Huang Z, Zhang W, Li Z. Detecting phishing activities in cryptocurrency networks using graph-based methods. IEEE Trans Comput Soc Syst. 2022;9(4):659-670.
4. Zhao M, Liu L, Yang F, *et al*. A hybrid model based on graph convolutional networks and deep learning for Ethereum phishing attack detection. Int. J Comput Sci Technol. 2024;13(2):111-119.
5. Kim H, Lee S, Choi D. Leveraging transformer networks for transaction graph-based cryptocurrency scam detection. Neural Comput Appl. 2024;35(7):123-134.
6. Yang Z, Zhou Z, Wang Q, *et al*. Blockchain-based money laundering detection using graph embedding techniques. Inf Sci. 2023;523:38-49.
7. Liu J, Sun T, Li Z, *et al*. A heterogeneous graph-based detection model for blockchain phishing accounts. J Inf Secur Appl. 2023;65(5):412-423.
8. Chen RTQ, Rubanova Y, Bettencourt J, *et al*. Neural ordinary differential equations for time-series modeling. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems; 2018 Dec 3-8; Massachusetts, USA. Cambridge (MA): MIT Press; c2018. p. 6572-6583.
9. Williams M, Bui D, Wang Y. GRU-ODE-Bayes: continuous modeling of sparse time-series. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019 Dec 8-14; Massachusetts, USA. Cambridge (MA): MIT Press; c2019. p. 7379-7390.
10. Zhang S, Zhao X, Zhang R. Traffic flow forecasting using dynamic graph convolutional networks and GRU-based models. J Transp Eng. 2023;45(3):221-230.
11. Xu Q, Wu L, Ma Y. Dynamic graph learning for financial fraud detection in blockchain transactions. J Comput Financ. 2023;28(4):543-552.