Real-Time Transaction Fraud Detection via Heterogeneous Temporal Graph Neural Network

Hang Nguyen[1,2][a] and Bac Le[1,2][b]
[1] *Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam*
[2] *Vietnam National University, Ho Chi Minh City, Vietnam*
fi

Keywords: Heterogeneous Graph Neural Network, Heterogeneous Temporal Graph Neural Network, Fraud Detection, Transaction Fraud Detection, Spatial-Temporal Graph Neural Network, Real-Time Fraud Detection.

Abstract: As digital transactions grow in prevalence, the threat of fraud has become a critical challenge for businesses and individuals. Fraudsters increasingly employ sophisticated tactics, disguising malicious activities as legitimate behavior, which renders traditional detection methods inadequate. This paper introduces a real-time fraud detection framework leveraging Heterogeneous Temporal Graph Neural Networks (HTGNN) to address these challenges. The proposed approach constructs a heterogeneous temporal graph from transaction data and employs a neural network architecture that integrates spatial, temporal, and semantic information. This allows for a comprehensive representation of transactions, entities, and their dynamic interactions over time. Unlike static approaches, our method captures the temporal evolution of behaviors, ensuring deeper insights into fraudulent patterns. The framework is designed to enhance detection accuracy while maintaining computational efficiency for real-time applications. Through rigorous experimentation and analysis, we expect to demonstrate that the proposed HTGNN framework significantly outperforms existing techniques in identifying fraudulent transactions, ultimately contributing to more robust and effective fraud detection systems.

1 INTRODUCTION

The shift to online operations, accelerated by digitalization and the COVID-19 pandemic, offers benefits like cost savings and accessibility but also heightens the risk of digital fraud across platforms like online banking, e-commerce, and social media. Such fraud leads to financial losses, data breaches, and eroded trust, driving urgent demand for advanced detection technologies. With the global fraud prevention market projected to reach $66.6 billion by 2028(Tumiwa et al., 2024), real-time fraud detection systems have become essential, enabling instant transaction monitoring to mitigate losses and protect customer trust.

However, the swift evolution of fraudulent activities poses significant challenges for current fraud detection and prevention methods, as demonstrated in studies such as (Awoyemi et al., 2017), (Zhou et al., 2019), (Dornadula and Geetha, 2019), (Maniraj et al., 2019), and (Sailusha et al., 2020). While traditional machine learning models are often used for identifying patterns in structured data, they often struggle to capture the complex and dynamic nature of real-world fraud scenarios. A particularly concerning trend is the rise of hidden or camouflaged behaviors, where malicious activities are deliberately masked to resemble legitimate transactions, making them harder to detect. To address these challenges, graph-based machine learning methods have emerged as promising solutions due to their ability to model intricate relationships and interactions between entities.

To address these limitations, graph-based machine learning methods have shown promise due to their ability to model the intricate relationships between entities and transactions. Graphs naturally represent complex dependencies by organizing data into nodes (representing entities such as users, transactions, and accounts) and edges (representing relationships between them). This structure enables the modeling of relationships that extend beyond simple transactions, allowing for a more nuanced understanding of fraud. In particular, heterogeneous temporal graphs are especially powerful in capturing the spatial, temporal, and semantic contexts of transactions and entities. These graphs can provide a rich, multifaceted

[a] https://orcid.org/0009-0004-7181-1406
[b] https://orcid.org/0000-0002-4306-6945

Nguyen, H. and Le, B.
Real-Time Transaction Fraud Detection via Heterogeneous Temporal Graph Neural Network.
DOI: 10.5220/0013161300003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 2*, pages 364-375
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

view of fraud, integrating diverse data types to better understand and detect malicious behavior. Temporal information allows systems to track the evolution of behaviors over time, spatial information reveals the interactions between different entities, and semantic data adds context to the meaning behind each transaction. Together, these aspects provide a comprehensive framework for fraud detection.

Despite significant advancements in graph-based algorithms, such as Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), and attention-based mechanisms (Rao et al., 2020), (Xiang et al., 2022), (Xiang et al., 2023a), and (Xie et al., 2023), most existing systems still focus on leveraging one or two of these dimensions-spatial, temporal, or semantic-often in isolation or in simple combinations. While these methods offer improvements over traditional techniques, they remain inadequate in capturing the full complexity of modern fraudulent activities. More critically, there is a lack of real-time systems that effectively integrate all three dimensions simultaneously, which is essential for timely fraud detection and prevention.

In this paper, we introduce a novel framework for real-time fraud detection that integrates spatial, temporal, and semantic information using heterogeneous temporal graphs within an inductive setting. Our approach addresses the limitations of current models by comprehensively analyzing fraud patterns across these three types of information while dynamically adapting to the evolving nature of fraudulent behaviors. By operating in real-time, our framework enhances the ability to detect and prevent fraudulent activities as quickly as possible, minimizing potential financial losses and mitigating risks before they escalate. The contributions of this paper include:

- We propose a methodology for transforming transaction data into a heterogeneous temporal graph structure, enabling the integration of spatial, temporal, and semantic information for learning transaction representations.
- We design a robust heterogeneous temporal graph neural network architecture and real-time system to effectively capture and learn comprehensive transaction representations.
- We conduct empirical experiments to evaluate the proposed framework on both real-world and synthesis datasets, demonstrating its effectiveness in accurately detecting fraud and efficiently handling real-time data processing.

The structure of this paper is as follows: Section 2 reviews the related work. Section 3 describes the details of the proposed method. Section 4 outlines the experimental setup. Section 5 presents the corresponding results and discussions. Section 6 concludes the paper and discusses potential future work.

2 RELATED WORK

In recent years, a wide range of machine learning techniques has been proposed to address the challenge of fraud detection. Early works primarily focused on traditional machine learning methods applied to real-world datasets. For instance, (Maes et al., 2002) utilized Bayesian Belief Networks (BBN) and Artificial Neural Networks (ANN) on a dataset obtained from Europay International, demonstrating the effectiveness of these models in identifying fraudulent credit card transactions. Similarly, (Sahin and Duman, 2011) employed decision trees and support vector machines (SVMs) on a major national bank's dataset, showcasing the potential of these methods in fraud detection tasks. Other studies, such as (Saputra et al., 2019), (Maniraj et al., 2019), (Dornadula and Geetha, 2019), (Sailusha et al., 2020), and (Varun Kumar et al., 2020), have explored ensemble learning methods to enhance detection accuracy. While these approaches achieved reasonable success, they primarily relied on static features and often lacked the ability to generalize well to more complex and dynamic fraud patterns.

With the rise of deep learning, researchers began exploring more sophisticated architectures to address the limitations of traditional machine learning models. Studies like (Fu et al., 2016), (Alarfaj et al., 2022), (Hasugian et al., 2023), and (Karthika and Senthilselvi, 2023) applied deep learning techniques, which outperformed earlier methods by learning intricate patterns in transaction data. However, these models typically focused on individual transactions or cardholders, thereby missing the broader context provided by the relationship between transactions and the entities involved. This limitation, highlighted by (Xiang et al., 2023b), indicated the need for models that can exploit both labeled and unlabeled data, especially in large-scale, real-world scenarios.

As fraud detection systems evolved, graph-based approaches began gaining attention due to their ability to model the complex relationships inherent in transactional data. The novel work by (Wang et al., 2019) and (Liu et al., 2020) laid the foundation for using graph neural network (GNN) in fraud detection, particularly for datasets with partial labels. These methods leveraged the relational structure of transactions to improve detection accuracy. Building on this, (Dou et al., 2020) and (Peng et al., 2021) introduced GNN-based techniques that incorporated reinforcement learning for neighbor selection, tackling the challenge of fraudsters' camouflage. Furthermore, (Liu et al., 2021) proposed PC-GNN, which effectively addressed the issue of imbalanced learning in graph-based fraud detection. These methods demonstrated that graph-based models could uncover patterns that traditional and deep learning models missed, particularly when it came to exploiting the relational data between transactions and entities.

One significant advancement in this domain was xFraud (Rao et al., 2020), which utilized a heterogeneous GNN architecture to aggregate transaction information and introduced a prediction result interpretation module, improving both model transparency and performance. (Xiang et al., 2022) proposed a joint feature learning framework for capturing spatial and temporal patterns in fraud detection, emphasizing the importance of modeling dynamic transaction behavior over time. In their subsequent work, (Xiang et al., 2023b) extended this by integrating entity information into transaction node representations and using temporal graph attention to aggregate historical transactions within a homogeneous graph structure. These studies highlight a growing trend towards more graph-based fraud detection models.

The BRIGHT framework, introduced by (Lu et al., 2022), represents one of the most significant attempts at developing a real-time fraud detection system. It leverages a Two-Stage Directed Graph (TD Graph) to enable efficient real-time inference by restricting message-passing to historical transaction data. This approach dramatically reduces computational overhead, making real-time detection feasible. Furthermore, BRIGHT employs the Lambda Neural Network (LNN) architecture to decouple the inference process into batch and real-time stages, improving both speed and accuracy. However, while BRIGHT makes strides in real-time detection, it focuses predominantly on spatial information using graph convolutional networks, which may not be robust enough to handle increasingly sophisticated fraud tactics. Despite the advancements, there remains a critical gap in the literature: the lack of comprehensive real-time fraud detection systems that fully leverage both temporal and semantic information surrounding transactions. Fraudsters continuously evolve their strategies, utilizing techniques such as obfuscation, transaction fragmentation, and exploiting security vulnerabilities, which current models may fail to capture adequately. Our proposal aims to build upon the BRIGHT framework by incorporating richer temporal and semantic data into the GNN-based fraud detection process. By integrating all historical time windows and capturing the contextual relationships between transactions and entities over time, our model seeks to enhance detection accuracy while maintaining the efficiency required for real-time applications.

3 METHODOLOGY

3.1 Heterogeneous Temporal Graph Construction

In this research, we address the problem of transaction fraud detection as a binary node classification task within an inductive setting on a heterogeneous temporal graph (HTG). Traditional fraud detection methods, such as rule-based systems or models that rely on static features, typically treat each transaction as an independent event. These methods often overlook the interconnected and dynamic nature of transactions over time, which can result in missed patterns or emerging fraud tactics. In reality, transactions are part of a broader ecosystem where relationships between entities (e.g., users, devices, locations) evolve, and fraud behaviors adapt. By failing to account for these evolving and interconnected factors, traditional approaches can struggle to detect sophisticated or hidden fraud. In contrast, our heterogeneous temporal graph model captures the intricate relationships between entities (e.g., cardholders, merchants, devices) and how these relationships evolve over time. This enables deeper insights into both normal and fraudulent behavior patterns, particularly in dynamic environments where fraud tactics constantly adapt.

The key challenge in building an effective fraud detection system lies in accurately modeling the representations of transactions. Each transaction is comprised of a diverse set of attributes, including quantitative data (e.g., transaction amount, timestamps) and categorical or identity-based data (e.g., cardholder information, merchant details, device identifiers). Basic encoding techniques like one-hot or label encoding are inadequate in capturing the complex, high-dimensional relationships between these attributes, leading to suboptimal model performance in detecting fraudulent patterns. To address this limitation, we propose the construction of a heterogeneous temporal graph (HTG), where various entities, such as cardholders, merchants, and devices, are represented as distinct node types. These node types interact with each other across time, allowing the model to learn more nuanced representations of how fraudulent behaviors evolve and how different entities interact.

Specifically, in the HTG, each node i is associated with a node type $\phi(i) \in \mathcal{A}$, where $\mathcal{A}$ is the set of all

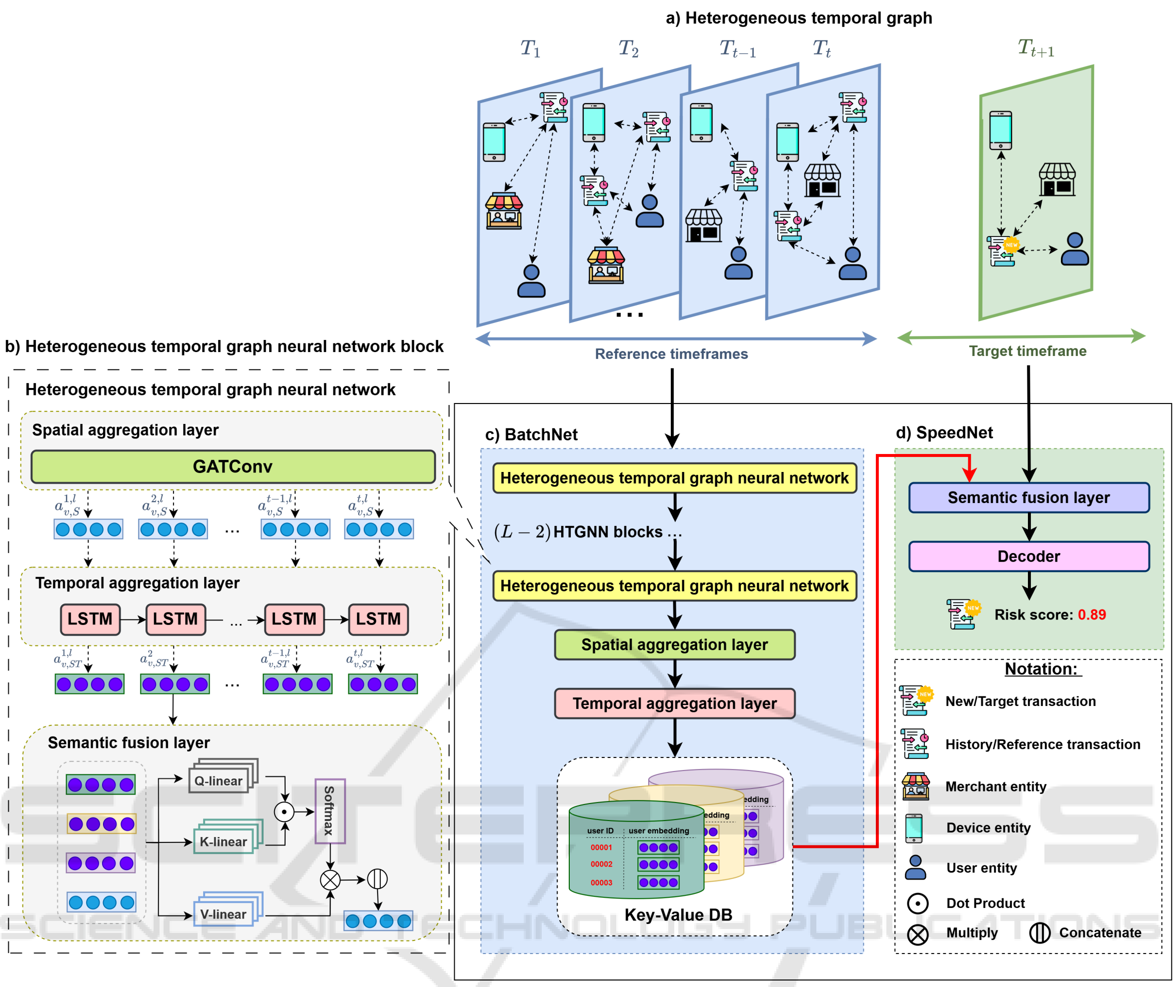

Figure 1: Proposed method: ***a)*** Illustration of a heterogeneous temporal graph: The graph depicts multiple timeframes labeled T_1, T_2,..., T_{t-1}, T_t, which represent the reference time periods. These timeframes contain historical transaction data and are used to generate insights from past interactions. The incoming timeframe T_{t+1} is highlighted as the target timeframe, containing all the transactions that require labeling. ***b)*** Depicting the HTGNN block components. ***c)*** Illustration of the architecture of Batch Net. ***d)*** Illustration of the architecture of Speed Net.

possible types of nodes, including transactions, cardholders, merchants, and devices. Depending on the business domain, additional entity types can be incorporated into the model, such as shipping addresses, payment tokens, or IP addresses, to enrich the graph structure and enhance detection accuracy. Edges $e \in \mathcal{E}$ are used to represent relationships between transaction nodes and these various entity nodes, forming a dynamic graph that evolves over multiple timeframes.

Temporal dynamics play a critical role in detecting fraud, as fraudulent activities often unfold gradually or are strategically hidden within patterns of legitimate behavior. To capture these evolving patterns, we introduce a temporal dimension to the graph by constructing it over multiple timestamps j, resulting in a temporal graph structure $\mathcal{G} = \{\mathcal{G}^j\}_{j=1}^{t+1}$, where $T = \{1, 2, \ldots, t-1, t\}$ denotes the reference time duration and $t+1$ denotes the timeframe that includes the new transactions needing classification, as illustration in Figure 1a. This temporal aspect allows the model to not only analyze relationships at a single point in time but also track how behaviors and interactions shift over time.

In our framework, we categorize all transaction nodes in a given timeframe $(t+1)$ as target nodes, which represent new transactions or transactions that need to be predicted. Other nodes are referred to as reference nodes. The relationship between target and reference nodes enables the model to leverage historical transaction data and identify abnormal patterns, even in new or previously unseen transactions. To facilitate the classification task, fraudulent transactions are labeled as 1, indicating they have been flagged or confirmed as fraudulent, while legitimate transactions are labeled as 0. The relationship between target and reference nodes enables the model to leverage historical transaction data and identify abnormal patterns, even in new or previously unseen transactions. The labels of transaction nodes are only used for reference nodes' original features during the training process without the risk of label leakage, ensuring the integrity of the classification task.

3.2 Heterogeneous Temporal Graph Neural Network (HTGNN)

Our approach stems from the observation that the transaction volume in the system increases significantly over time, while each transaction remains unique and lacks distinct temporal features. This poses a challenge: extracting temporal information from all transaction nodes can lead to resource inefficiencies and reduced performance, as each user's transaction will be different depending on their behavior. To address this issue, we propose to shift the focus to entity nodes, which are more representative of meaningful patterns in the data. Transactions exhibit clear sequential characteristics, occur at specific timestamps, and are often linked to previous and subsequent transactions through shared entity attributes, such as user identity, device type, or IP address. By focusing on these entity nodes, we can efficiently extract temporal information directly linked to historical transactions associated with each entity. In this context, we designed a heterogeneous temporal graph neural network block with multiple layers, including a spatial aggregation layer, temporal aggregation layer, and semantic fusion layer, as in Figure 1b. In the following subsections, we will provide a detailed architecture of the layers within the l-th HTGNN block.

3.2.1 Spatial Aggregation Layer

In the proposed graph construction method, entity nodes (e.g., cardholders, merchants, devices) are connected to transaction nodes across multiple timeframes. To effectively represent these entities, we compute spatial embedding vectors by aggregating information from historical transactions associated with them. However, not all are equally informative for the entity's final representation in the model. To address this, we leverage a Graph Attention Network (GAT) (Veličković et al., 2017), which assigns attention coefficients that determine the importance of each transaction in relation to the entity node. This approach helps to prioritize relevant transactions and reduce the influence of noisy or less significant ones.

Each entity node $v \in \mathcal{A}'$ (where $\mathcal{A}'$ is the set of entity types, $\mathcal{A}' \subset \mathcal{A}$) is associated with a spatial embedding that evolves over time. The embedding of node v at timestamp t in the l-th HTGNN block is computed by aggregating information from neighboring transaction nodes that occur at timestamp t. Specifically, at each timestamp t and in the l-th HTGNN block, the spatial embedding of an entity node v is computed as follows:

$$a_{v,S}^{t,l} = GATConv_{\phi(v)}(a_u^{t,l-1} : u \in \mathcal{N}^t(v))$$

Here, $a_{v,S}^{t,l} \in \mathbb{R}^{d_{trans}}$ represents the spatial embedding of entity node v at timestamp t in the l-th HTGNN block, with d_{trans} being the dimension of the transaction node embedding. $\mathcal{N}^t(v)$ denotes the set of neighboring transaction nodes at timestamp t, and $a_u^{t,l-1} \in \mathbb{R}^{d_{trans}}$ is the embedding of transaction node u at the previous block $(l-1)$. The initial embedding $a_u^{t,0}$ is set to the raw feature vector of transaction node u at timestamp t. $GATConv_{\phi(v)}$ refers to the graph attention layer specific to the type of entity node v. The aggregation of information from neighboring transactions for entity node v is expressed as:

$$a_{v,S}^{t,l} = \sum_{u \in \mathcal{N}^t(v)} \alpha_{v,u} \mathrm{W_u} a_u^{t,l-1}$$

where the attention coefficients $\alpha_{v,u}$, representing the importance of transaction node u to entity node v, are computed as:

$$e(u,v) = \left(\mathrm{LReLU}\left(\mathbf{a}_v^\top \mathbf{W}_{\phi(v),v} \mathbf{a}_v^{t,l-1} + \mathbf{a}_u^\top \mathbf{W}_u \mathbf{a}_u^{t,l-1}\right)\right)$$

$$\alpha_{u,v} = \frac{\exp(e(u,v))}{\sum_{k \in \mathcal{N}^t(v)} \exp(e(k,v))}$$

Here, $e(u,v)$ represents the score for the relationship between nodes u and v. $\mathbf{W}_{\phi(v),v}$ and $\mathbf{W}_u$ are learnable linear transformations that map the feature vectors of entity node v and transaction node u into the same space. The activation function used is LeakyReLU, and $\mathbf{a}_v$ and $\mathbf{a}_u$ are learnable attention weight vectors specific to the entity and transaction nodes, respectively. Different transformation matrices, $\mathbf{W}_{\phi(v),v}$, are applied to different types of entity nodes.

To ensure robustness and improve the model's ability to learn from complex graph data, we apply multi-head attention. This technique runs multiple attention mechanisms in parallel and aggregates their outputs, providing a more stable and reliable representation for each entity node. Moreover, the transaction nodes act as source nodes, providing the raw information, while the entity nodes act as target nodes, whose spatial embeddings are updated based on the aggregated transaction data. Importantly, we do not include self-loops during aggregation, as the focus is on the aggregation of transactions around them at the specific timestamp. Finally, we concatenate the raw entity features at timestamp t, a_v^t, to form the complete spatial embedding, as:

$$a_{v,S}^{t,l} = CONCATENATE(a_{v,S}^{t,l}, a_v^t)$$

In our experiments, raw entity features are constructed from several time-varying attributes. For instance, for a user entity node, these features include the total number of recorded fraudulent transactions, the number of distinct IP addresses used, the average transaction amount, and more. These dynamic features capture the evolving behavior of entities over time, enabling the model to represent temporal patterns more effectively. The spatial embeddings of entity nodes at each timeframe t are then combined across multiple timestamps to generate temporal embeddings, forming a comprehensive spatial-temporal representation for the entity node.

3.2.2 Temporal Aggregation Layer

After computing the spatial embeddings for each entity node at individual timestamps, we move to the temporal embedding phase. Temporal embeddings are crucial in fraud detection, as they enable the model to capture the evolving behavior of entities over time, identifying patterns that may indicate fraudulent activity. Once the spatial embeddings for entity nodes are obtained across timeframes, we combine them from timeframe 1 to timeframe t to derive the spatial-temporal embeddings. This process not only captures the evolving relationships and interactions among entities but also provides deeper insight into their historical context.

The core idea is to maintain a persistent memory of previous time windows, allowing the model to retain historical information as it progresses through timeframes. To achieve this, we employ a Long Short-Term Memory (LSTM) network, which is well-suited for capturing temporal dependencies by maintaining long-term memory through its hidden state and cell state.

Specifically, after constructing the spatial embeddings for each entity node at each timeframe t, we used these embeddings to represent temporal dependencies. For each entity node v, we combine spatial embeddings from timeframes 1 to t to generate a comprehensive spatial-temporal embedding. The LSTM model is used to process these sequential embeddings:

$$a_{v,ST}^{t,l} = LSTM_{\phi(v)}(a_{v,S}^{1,l}, a_{v,S}^{2,l}, \dots, a_{v,S}^{t,l})$$

where $a_{v,S}^{t,l}$ represents the spatial embedding of entity node v at time t and HTGNN block l-th, and $a_{v,ST}^{t,l}$ denotes the spatial-temporal embedding for node v that captures information across all time windows up to t. $LSTM_{\phi(v)}$ refers to the LSTM layer specific to the type of entity node v.

3.2.3 Semantic Fusion Layer

In this module, we focus on extracting the contextual feature of a transaction node u at timestamp t in the HTGNN block l-th by the mutual information from its neighboring entity nodes at timestamp $t-1$ in the HTGNN block l-th, as well as from its own features, to create a comprehensive embedding for it. Before fusing the semantic information, we project the spatial-temporal embedding vector of each neighboring entity node v at timestamp $t-1$ in the HTGNN block l-th and the transaction's feature vector into the same latent space. This ensures the vectors representing entities and transactions are the same dimensions.

The fusion of semantic information is achieved by applying scaled dot-product attention, inspired by the Transformer model (Vaswani et al., 2017), which is well-suited for capturing complex, multi-entity interactions. For a given transaction node u, the spatial-temporal embedding vectors of its neighboring entity nodes are aggregated into a list, denoted as:

$$E := [a_u^{t,l}, a_{v,ST}^{t-1,l}] \;\forall v \in \mathcal{N}^t(u) \text{ and } \phi(v) \in \mathcal{A}'$$

where n is the number of entity types; $a_{u,\mathcal{A}_0}^{t,l}$ is the feature vector of the transaction; and $a_{v,ST}^{t-1,l}$ represent all the spatial-temporal embedding of all neighbor entity node of node u at timestamp $t-1$ in the HTGNN block l-th. This setup allows us to compute the semantic relationship between the transaction and its neighboring entity nodes. To capture the relationships between these node types, we apply the following procedure:

(1) Transform all vectors in the list E to a query, key, and value vector.

$$q_{\mathcal{A}_i} = W_{query} \cdot E[i], \forall i = 0 \to n$$
$$k_{\mathcal{A}_i} = W_{key} \cdot E[i], \forall i = 0 \to n$$
$$v_{\mathcal{A}_i} = W_{value} \cdot E[i], \forall i = 0 \to n$$

(2) Compute mutual attention weight by the dot product between the query and key vector.

$$\delta(\mathcal{A}_i, \mathcal{A}_j) = \frac{\exp([q_{\mathcal{A}_i}]^T \cdot [k_{\mathcal{A}_j}])}{\sum_{j=0}^{n} \exp([q_{\mathcal{A}_i}]^T \cdot [k_{\mathcal{A}_j}])}$$

(3) The mutual information between node i and node j is represented as a weighted sum of all value

vectors $v_{\mathcal{A}_j}$ and the computed mutual attention value $\delta(\mathcal{A}_i, \mathcal{A}_j)$. This can be expressed mathematically as:

$$s_{\mathcal{A}_i} = \sum_{j=0}^{n} \delta(\mathcal{A}_i, \mathcal{A}_j) \cdot v_{\mathcal{A}_j}$$

Where $W_{query}, W_{key}, W_{value} \in \mathbb{R}^{d \times d}$ and θ are the learnable parameters shared across all entity node types. The final embedding of transaction u is created by concatenating those outputs and using W_{trans} as the linear transformation to ensure the length of the transaction node feature vectors' dimension is maintained through many layers, formulated as:

$$a_u^{t,l} = [s_{\mathcal{A}_0} || s_{\mathcal{A}_1} || s_{\mathcal{A}_2} || \dots || s_{\mathcal{A}_n}]$$
$$a_u^{t,l} = W_{trans} \cdot a_u^{t,l}$$

In short, $a_u^{t,l}$ is the output of the HTGNN block l-th. By enhancing the message-passing process, HTGNN integrates both spatial and temporal dimensions, moving beyond the simplistic aggregation methods of conventional GNNs. The use of transformer-based architectures for updating allows HTGNNs to effectively capture the contextual information, leading to richer node representations and improved performance in the fraud detection system.

3.3 Real-Time Transaction Fraud Detection System

While the stacked HTGNN blocks can provide high accuracy in offline fraud detection by capturing complex spatial, temporal, and semantic information, their computational complexity makes them less suitable for real-time inference. Running full-batch HTGNN models in a real-time setting would introduce significant latency, delaying fraud detection when immediate decisions are required. To address this challenge, we implement a dual-model approach based on Lambda architecture, which enables the system to maintain both accuracy and efficiency by separating the process into batch and speed layers.

The batch layer is designed to handle large-scale, high-complexity computations offline. In this layer, the stacked HTGNN blocks continuously process historical transaction data to detect emerging fraud patterns and update the model periodically. This ensures that the model remains accurate by learning from new data and adjusting to evolving fraudulent behaviors. In this layer, we design the BatchNet to learn the embedding of all reference entity nodes. BatchNet consists of **L** stacked HTGNN blocks, one spatial aggregation layer, and one temporal aggregation layer, as in figure 1c. Each HTGNN block allows transaction nodes to gather mutual information from their own features as well as spatial-temporal embeddings of related entities at preceding timestamps. The stacked architecture enhances message passing between blocks, allowing nodes to aggregate information from further away in the graph, which enhances their representations. Instead of utilizing all HTGNN block at the end of BatchNet, we employ only the two first layers to extract the spatial-temporal embedding of entities, which are then stored as entity embeddings in key-value databases for future use. We choose the key-values database to store spatial-temporal of entities nodes to optimize real-time fraud detection by enabling quick retrieval, reducing database load, and improving memory efficiency, which is crucial for handling large datasets and ensuring rapid decision-making.

To ensure real-time responsiveness, we designed SpeedNet for the speed layer, a lightweight model focused on minimizing prediction latency. This layer operates on streaming transaction data and leverages the precomputed embeddings of all related entities generated by the BatchNet. By utilizing these precomputed features, the SpeedNet can make rapid predictions without having to process the entire graph structure for each incoming transaction in timestamp $t+1$. SpeedNet, illustrated in figure 1d, is built with a semantic fusion layer that extracts mutual information from the features of the target transaction at timestamp $t+1$ and the corresponding entity embeddings from timestamp t queried from the key-value database. This process generates the final embedding for the target transaction, which is then used to compute the transaction's risk score by the decoder layer. In our environment, we used two multi-perceptron layers and a softmax function. By combining these two layers, the system strikes a balance between accuracy and efficiency. The batch layer ensures the model remains effective by continuously learning from past transactions, while the speed layer ensures that real-time transactions are processed quickly enough to meet the demands of a live fraud detection environment.

3.4 Training Process

In the training process of our fraud detection system, both BatchNet and SpeedNet are trained across multiple time windows. This training method involves sampling data from specific time intervals to create mini-batches. Specifically, we consider all historical data from timestamp 1 to t_{latest} for training, in which t_{latest} is the latest timestamp. However, due to the extensive nature of this interval, which can result in a large volume of data, we divide it into smaller, manageable time windows for training. Each mini-batch corresponds to a time partition denoted as $T = \{1, 2, \ldots, t\}$, where each unit represents a timestamp, such as minutes, hours, or days. In our experiments, we define each timestamp as a day. Specifically, the reference subgraph covering timestamps from 1 to $t-1$, denoted as $\{\mathcal{G}^j\}_{j=1}^{t-1}$, serves as the input for BatchNet. In contrast, the target subgraph at timestamp t, along with the spatial-temporal embeddings of all entities at timestamp $t-1$, is provided as input to SpeedNet.

Following the completion of training at time window T, the sliding window mechanism comes into play. We incrementally shift the window to the next unit, adjusting it to $T = 2, 3, ..., t+1$. This step allows us to integrate the most recent transactions into the training set while continuing to leverage the historical data encapsulated in previous timestamps. The sliding window ensures that the model is exposed to new transaction patterns over time, enabling it to adapt to evolving fraud behavior. Moreover, instead of retraining from scratch with each new window, we initialize the model with the parameters learned in the previous training iteration. This transfer of knowledge is crucial for refining the model's understanding of both short-term and long-term fraud dynamics, as it incrementally builds upon insights gained from earlier windows. By doing so, the model not only achieves better generalization but also efficiently handles large-scale datasets that span extended periods.Not only that, throughout the training process, the labels of all transaction nodes in the target subgraph at timestamp t are used to compute the loss function, enabling the model to update its predictions. The loss is calculated using a binary cross-entropy function:

$$\mathcal{L} = -\sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right)$$

Where y_i represents the ground truth label (fraudulent or normal) of transaction node i, and $\hat{y}_i$ denotes the risk score of node i being fraudulent. By iteratively minimizing this loss function across multiple sliding windows, our system continuously improves its ability to detect fraudulent transactions, even as new data arrives.

4 EXPERIMENTS

4.1 The Usage Datasets

We use a real-world transaction dataset, the IEEE CIS Fraud Detection dataset (referred to as the Vesta dataset), which was released for the IEEE CIS Fraud Detection competition. This comprehensive dataset includes 590,540 transaction records, each meticulously labeled as either normal or fraudulent. Of these, 20,663 transactions are fraudulent, comprising approximately 3.5% of the dataset. The dataset contains detailed information on transaction amounts, payment methods, and device information, with 433 attributes available for each transaction, offering a rich and diverse set of features for building machine learning models for fraud detection. In addition to the transaction labels, the IEEE dataset records transaction information over time, spanning a 6-month period.

In addition to using real-world datasets, we conduct further experiments using synthetic data. The PAYSIM dataset is a synthetic dataset generated from simulations that replicate real-world financial transactions. It was designed to address privacy concerns while retaining the statistical properties and transactional behaviors observed in natural financial systems. This dataset is created by a sophisticated financial simulator based on real transaction data. The PAYSIM dataset includes 6,362,620 transactions, of which 8,213 are fraudulent (the frau ratio approximately 0.129%) each labeled as either normal or fraudulent. In addition to the transaction labels, this dataset records transaction information over time, covering a 1-month period. The dataset provides detailed attributes for each transaction, such as time step, type, amount, sender name, receiver name, and old and new balances for both sender and receiver. A summary of the training and testing data statistics for both datasets is presented in table 1.

Table 1: Training and Testing Data Statistics. Note: #Timeframes is the number of timeframes used in the training and testing phase. Similarly, #Transactions is the number of transactions used.

Dataset	**#Timeframes**		**#Transactions**	
	Train	**Test**	**Train**	**Test**
CSI	145	36	487,837	102,703
Paysim	24	6	6,239,040	123,580

4.2 Baseline

To evaluate the performance of our proposed method, we compare it against several established baseline models in fraud detection, including traditional machine learning algorithms, graph-based methods. Below, we outline the baseline models used in our experiments:

- **LightGBM (LGB):** is a fast, decision-tree-based algorithm designed for large-scale datasets, making it ideal for fraud detection. It handles both categorical and continuous features efficiently, supports imbalanced datasets, and offers quick training times.
- **BRIGHT**(Lu et al., 2022)**:** is a proposed solution designed to tackle challenges in deploying Graph Neural Networks (GNNs) for real-time fraud detection. It utilizes historical transaction data to derive insights for new transactions through a Graph Convolutional Network (GCN). This method models relationships between transactions and entities, allowing the network to aggregate relevant information from past transactions.

4.3 Implement Details

4.3.1 Feature Encoding

In our experiments, we begin by preprocessing each dataset to create a clean version. We either retain the original values or transform them based on our understanding of the relevant business domain, while also encoding categorical attributes. Consequently, each row in the feature matrix corresponds to a transaction. For the LightGBM (LGB) model, we directly use this feature matrix as input. In contrast, for graph-based methods, we apply normalization before utilizing the features. For BRIGHT, we also incorporate this matrix for the transaction node features in the input graph, while setting the features of the entity nodes to zero vectors, as suggested in the original proposal. For our proposed approach, we selectively extract components from the feature matrix to represent the transaction node features, with entity node features derived from the same matrix but tailored to reflect the unique characteristics of each entity, as shown in Figure 2.

4.3.2 Experimental Setup

To identify the optimal hyperparameters, we utilize grid search to determine the ideal configuration for the graph neural network (GNN), specifically focusing on the number of layers and hidden units. We

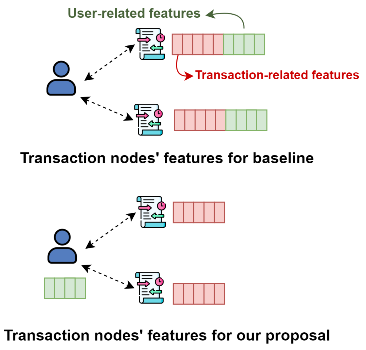
User-related features
Transaction-related features
Transaction nodes' features for baseline
Transaction nodes' features for our proposal

Figure 2: Illustration of the difference features are used in experiments.

evaluate GNN layer counts from the set $\{1, 2, 3, 4, 6, 8, 16\}$ and hidden unit options from $\{56, 128, 256, 512\}$. For the LightGBM model, we train it using 10,000 trees. We incrementally adjust the number of HTGNN blocks and time window size, selecting configurations that optimize model performance. For the training process, we employ the Adam optimizer with a learning rate of 10^{-3} and a weight decay of 10^{-4}. The remaining hyperparameters include a dropout rate of 0.2, a total of 10,000 epochs, and an early stopping mechanism with a patience of 50 epochs. All experiments are conducted on a DGX server equipped with four A100 GPUs.

4.4 Evaluation Metrics

In evaluating our proposed fraud detection method, we use the following metrics:

- **Average Precision (AP)** evaluates binary classification, especially on imbalanced data, by averaging precision and recall over thresholds.
- **AUC-ROC** measures a model's ability to distinguish classes, showing the balance between true and false positives across thresholds.
- **Prediction time (or Latency)** is the time a model takes to predict after input. Low latency is critical for timely fraud detection.

5 RESULTS AND DISCUSSION

5.1 Experimental Results

The comparative analysis of fraud detection methods reveals the superior performance of the HTGNN 2-hop model across both the IEEE-CIS and PAYSIM datasets, as shown in Table 2 and Figure 3. On the IEEE-CIS dataset, the HTGNN 2-hop achieved the highest AUC-ROC score of 0.94, significantly outperforming traditional models like LGB, which recorded

Table 2: The comparative performance of different models for fraud detection on IEEE-CIS Fraud Detection and PAYSIM dataset. Notes: the prediction time is measured in milliseconds.

Method	IEEE-CIS			PAYSIM		
	AUC-ROC	AP	Prediction time	AUC-ROC	AP	Prediction time
LGB	0.851±0.003	0.361±0.025	**0.69±0.019**	0.882±0.013	0.454±0.011	**0.52±0.019**
BRIGHT	0.863±0.014	0.371±0.012	1.81±0.014	0.891±0.021	0.475±0.005	1.67±0.01
HTGNN 1-hop	0.931±0.013	0.611±0.007	1.85 ± 0.007	0.923±0.023	0.662±0.002	1.68±0.012
HTGNN 2-hop	**0.940±0.010**	**0.641±0.008**	1.80±0.013	**0.956±0.003**	**0.682±0.011**	1.67±0.041
HTGNN 3-hop	0.896±0.004	0.591±0.007	1.87 ± 0.001	0.907±0.003	0.623±0.003	1.69 ± 0.006

an AUC-ROC of 0.85, and BRIGHT, which scored 0.86. Additionally, the HTGNN 2-hop's AP score of 0.64 far exceeded that of LGB 0.36 and BRIGHT 0.37, underscoring its enhanced capability in identifying true fraud cases.

On the PAYSIM dataset, the HTGNN 2-hop again stood out with an impressive AUC-ROC of 0.96, outperforming both LGB 0.88 and BRIGHT 0.89. Its AP score of 0.68 on PAYSIM further reinforced its superiority, surpassing LGB's 0.45 and BRIGHT's 0.48. While the HTGNN 2-hop model required slightly longer training times than the traditional models, its prediction time of 1.8 milliseconds on IEEE-CIS and 1.67 milliseconds on PAYSIM remained comparable to that of BRIGHT and manageable for real-time deployment.

These results highlight that, while traditional models like LGB and BRIGHT offer faster prediction times, the HTGNN 2-hop model provides a substantial boost in both AUC-ROC and AP, making it the most effective method for capturing complex fraud patterns and improving detection accuracy. As depicted in Figure 4, the histogram of risk scores for normal transactions reveals that our proposed HTGNN 2-hop model concentrates the majority of normal transaction scores below 0.5 on both datasets. In contrast, both the LGB and BRIGHT models show a significantly higher number of normal samples with risk scores above 0.5, indicating a higher likelihood of misclassifications. This difference highlights that the HTGNN 2-hop model generates more reasonable and accurate risk scores, thereby reducing the chance of false positives in fraud detection. The ability to push genuine transactions into lower risk score ranges is a clear indicator of the model's superior calibration, further solidifying its reliability in high-stakes fraud detection environments. Overall, the HTGNN 2-hop model demonstrates a commendable balance between performance and efficiency, establishing it as a leading approach for fraud detection tasks within heterogeneous temporal graphs.

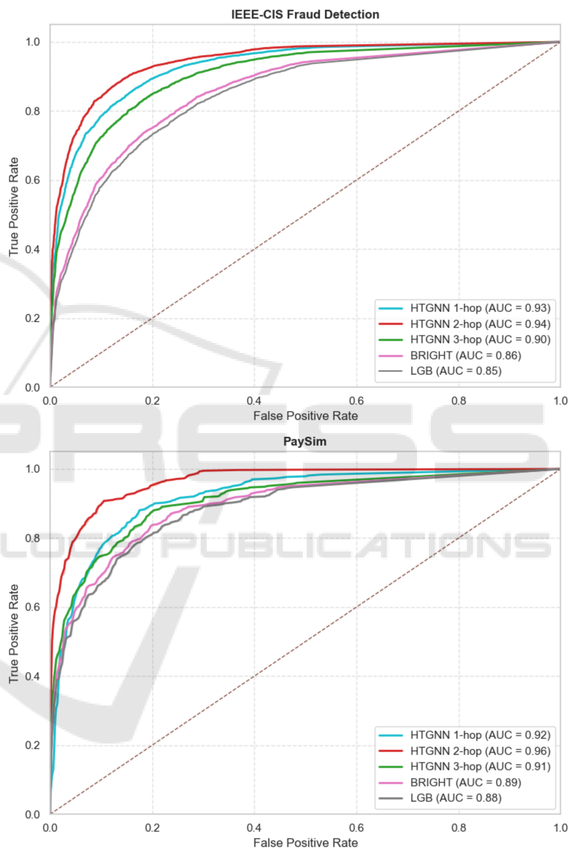

Figure 3: Receiver operating characteristic curve.

5.2 Discussion on the Application of HTGNN to Real Fraud Detection Systems

Training HTGNN across various product scenarios offers both opportunities and challenges, especially in data representation and feature extraction. With heterogeneous data and transaction timelines, HTGNN efficiently captures relationships and temporal dynamics. It adapts to fields like e-commerce, insurance, and telecommunications. For example, in

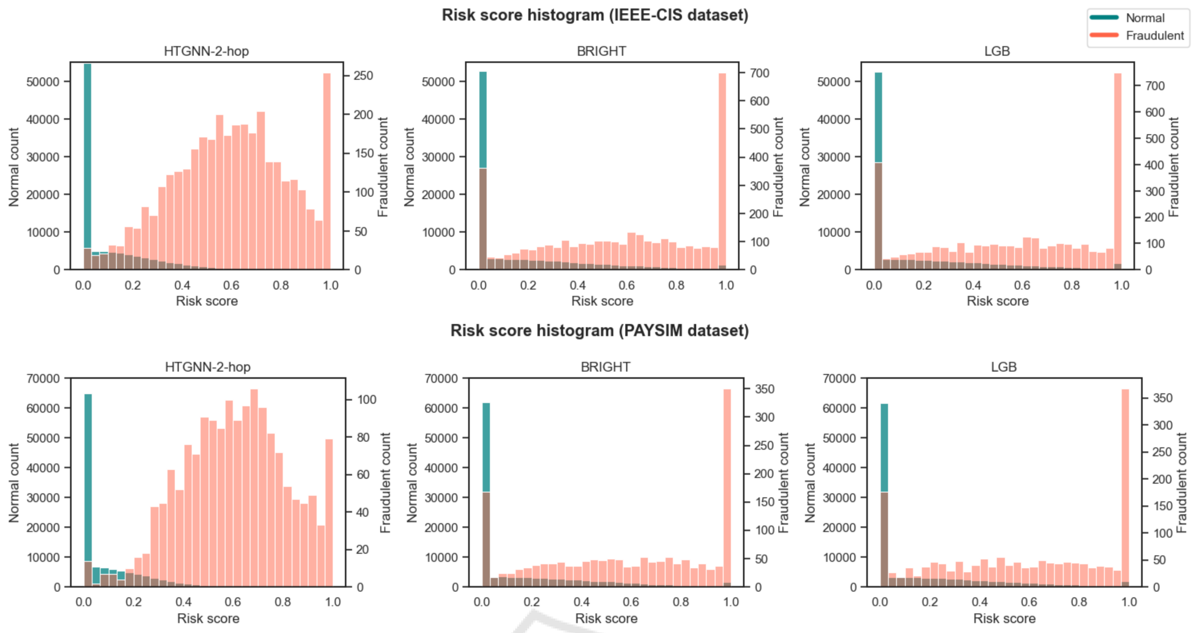

Figure 4: The risk score histogram.

e-commerce, HTGNN can identify fraudulent activities by analyzing user behavior patterns, such as sudden changes in purchase frequency, unusually large transactions, or abnormal browsing habits. However, it struggles to predict transactions involving unseen entities, as it assigns them a zero vector, limiting accuracy due to the lack of historical data.

6 CONCLUSION

In this paper, we introduced a novel framework for real-time fraud detection that leverages the power of heterogeneous temporal graphs to integrate spatial, temporal, and semantic information. By utilizing a robust heterogeneous temporal graph neural network (HTGNN) architecture, our approach captures complex relationships and evolving patterns of fraud that traditional models often miss, particularly those related to hidden or disguised fraudulent activities. Our framework operates in real-time, enabling early detection of fraudulent transactions, thereby minimizing financial losses and reducing operational risks for organizations. The empirical results from our evaluations on large, complex datasets demonstrate the effectiveness of the proposed model in accurately detecting fraud and handling real-time data processing. This work provides a significant advancement in the field by offering a comprehensive and adaptive solution to the challenges posed by evolving fraud tactics. Future work could explore extending this framework by incorporating more sophisticated graph-based models, with a focus on enhancing interpretability and providing clearer insights into the decision-making process.

ACKNOWLEDGEMENT

This research was partially funded by the Vingroup Innovation Foundation (VINIF) under the grant number VINIF.2021.JM01.N2.

REFERENCES

Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., and Ahmed, M. (2022). Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. *IEEE Access*, 10:39700–39715.

Awoyemi, J. O., Adetunmbi, A. O., and Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 international conference on computing networking and informatics (ICCNI)*, pages 1–9. IEEE.

Dornadula, V. N. and Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. *Procedia computer science*, 165:631–641.

Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., and Yu, P. S. (2020). Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 315–324.

Fu, K., Cheng, D., Tu, Y., and Zhang, L. (2016). Credit card fraud detection using convolutional neural networks. In *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part III 23*, pages 483–490. Springer.

Hasugian, L. S. et al. (2023). Fraud detection for online interbank transaction using deep learning. *Journal of Syntax Literate*, 8(6).

Karthika, J. and Senthilselvi, A. (2023). Smart credit card fraud detection system based on dilated convolutional neural network with sampling technique. *Multimedia Tools and Applications*, 82(20):31691–31708.

Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., and He, Q. (2021). Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*, pages 3168–3177.

Liu, Z., Dou, Y., Yu, P. S., Deng, Y., and Peng, H. (2020). Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1569–1572.

Lu, M., Han, Z., Rao, S. X., Zhang, Z., Zhao, Y., Shan, Y., Raghunathan, R., Zhang, C., and Jiang, J. (2022). Bright-graph neural networks in real-time fraud detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3342–3351.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit card fraud detection using bayesian and neural networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, volume 261, page 270.

Maniraj, S., Saini, A., Ahmed, S., and Sarkar, S. (2019). Credit card fraud detection using machine learning and data science. *International Journal of Engineering Research*, 8(9):110–115.

Peng, H., Zhang, R., Dou, Y., Yang, R., Zhang, J., and Yu, P. S. (2021). Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–46.

Rao, S. X., Zhang, S., Han, Z., Zhang, Z., Min, W., Chen, Z., Shan, Y., Zhao, Y., and Zhang, C. (2020). xfraud: explainable fraud transaction detection. *arXiv preprint arXiv:2011.12193*.

Sahin, Y. and Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 1–6.

Sailusha, R., Gnaneswar, V., Ramesh, R., and Rao, G. R. (2020). Credit card fraud detection using machine learning. In *2020 4th international conference on intelligent computing and control systems (ICICCS)*, pages 1264–1270. IEEE.

Saputra, A. et al. (2019). Fraud detection using machine learning in e-commerce. *International Journal of Advanced Computer Science and Applications*, 10(9).

Tumiwa, R. A. F., Purba, J. H. V., Zaroni, A. N., and Judijanto, L. (2024). Management of antifraud in the era of banking digitization. *International Journal*, 5(10):2355–2367.

Varun Kumar, K., Vijaya Kumar, V., Vijay Shankar, A., and Pratibha, K. (2020). Credit card fraud detection using machine learning algorithms. *International journal of engineering research & technology (IJERT)*, 9(7):2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

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

Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., Yu, Q., Zhou, J., Yang, S., and Qi, Y. (2019). A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE international conference on data mining (ICDM)*, pages 598–607. IEEE.

Xiang, S., Cheng, D., Shang, C., Zhang, Y., and Liang, Y. (2022). Temporal and heterogeneous graph neural network for financial time series prediction. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pages 3584–3593.

Xiang, S., Zhu, M., Cheng, D., Li, E., Zhao, R., Ouyang, Y., Chen, L., and Zheng, Y. (2023a). Semi-supervised credit card fraud detection via attribute-driven graph representation. In *AAAI Conference on Artificial Intelligence*.

Xiang, S., Zhu, M., Cheng, D., Li, E., Zhao, R., Ouyang, Y., Chen, L., and Zheng, Y. (2023b). Semi-supervised credit card fraud detection via attribute-driven graph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14557–14565.

Xie, Y., Liu, G., Zhou, M., Wei, L., Zhu, H., and Zhou, R. (2023). A spatial-temporal gated network for credit card fraud detection. In *2023 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, volume 1, pages 1–6. IEEE.

Zhou, X., Zhang, Z., Wang, L., and Wang, P. (2019). A model based on siamese neural network for online transaction fraud detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7.