TRUST: Transaction Risk via Unified Sequence and Topology

Rithvik Y, Bhavuk Singhal, Shubham Jain, Akshat Garg, Karan Tanwar, Anshu Aditya, Debashis Mukherjee, Debdoot Mukherjee

Meesho, India

{rithvik.y, bhavuk.singhal, shubham.jain1, akshat.garg, karan.tanwar, anshu.aditya, debashis.mukherjee, debdoot.mukherjee}@meesho.com

Abstract

Abuse detection in e-commerce platforms is critical for preventing operational losses, particularly for transaction types vulnerable to abuse such as Return-to-Origin (RTO) in Cash-on-Delivery (COD) workflows. Detecting such abuse accurate, real-time decisions to intercept malicious orders before placement, imposing stringent sub-second latency requirements on deployed systems. In this work, we present **TRUST**, a deployed, production-scale abuse detection system based on a unified architecture of heterogeneous Graph Neural Networks (GNNs) and Transformer-based sequence encoders. This design enables joint reasoning over multi-relational entity interactions and temporal behavioral signals, allowing the model to combine complementary information for effective abuse detection when either modality is sparse or absent. TRUST processes millions of transactions daily with an average inference latency of $\sim$25 ms, achieving a $\sim$9.6% absolute precision improvement over a strong XGBoost baseline in live RTO detection. We report systematic ablation studies across both graph and sequence stages, evaluating GNN variants, sampling strategies, sequence lengths, and positional encoding schemes to guide architectural choices. Deployed end-to-end in a high-throughput environment, TRUST demonstrates that GNN–Transformer cascades can deliver state-of-the-art accuracy, scalability, and operational reliability in real-world abuse detection, offering a reproducible blueprint for similar industry-scale applications.

1 Introduction and Background

Cash-on-Delivery (COD) is the most popular payment method in Indian e-commerce, accounting for 60–65% of orders. Although only a small portion of these orders end in Return-to-Origin (RTO)—where shipments are returned due to fake addresses, cancellations, or refusals—their financial and operational impact is significant (Dazeinfo 2024). Each RTO drives direct logistics costs and indirect impacts on inventory and service operations. The financial implications are substantial: global online payment fraud losses are projected to grow from $48 billion in 2023 to over $107 billion by 2029, with cumulative losses of $343 billion between 2023 and 2027 (Juniper Research 2024). In the Indian context, the challenge is particularly acute. The LexisNexis Risk Solutions Asia Pacific Study reports that for every ₹1 lost to fraud, businesses suffer a total cost of ₹4, once operational overheads, chargebacks, and customer service costs are included (LexisNexis 2024).

Unlike payment fraud in financial services, RTO abuse disrupts the entire commerce pipeline—sellers, logistics networks, and customers. Even small fractions of fraudulent activity erode margins in logistics-heavy models, highlighting the need for scalable, interpretable, and real-time RTO detection. There are multiple challenges to be addressed in RTO prediction task:

- **Data imbalance:** RTO prone orders are rare, risking overfitting to majority behavior
- **Evolving fraud:** Fraudsters adapt via bots, synthetic identities, and collusion among various entities
- **Entity heterogeneity:** Signals span users, suppliers, products, and logistics, requiring multi-relational modeling.
- **Hidden risk:** Seemingly clean users may be structurally linked to RTO. Fraudulent Users does some clean orders initially to build a good profile only to commit RTO later.
- **Sparse histories:** New or low-activity users lack dense behavioral data.
- **Temporal volatility:** RTO emerges through sudden shifts in returns and orders.

To address these challenges, we present a novel unified RTO detection system tailored for RTO risk mitigation in high-volume e-commerce platforms. Our key contributions are as follows:

1. We propose a novel unified pipeline that first encodes user–supplier relationships using GNN, followed by temporal behavior modeling using Transformer-based sequence encoders. This design enables the system to jointly reason over structured, topological, and sequential RTO signals in high-dimensional, sparse environments.
2. We conducted targeted experiments addressing key RTO-specific challenges, and performed extensive ablation studies to guide architectural choices across both GNN and Transformer components.
3. We deployed TRUST end-to-end in a high-throughput production environment, where it achieved a $\sim$9.6% precision gain over a strong XGBoost baseline. Our deployment experience highlights both the system's real-world

effectiveness and practical lessons for operating RTO detection systems at scale.

2 Related Works

Internet fraud has plagued online commerce since its inception, fueling distrust and prompting ongoing efforts to detect and prevent it. Early detection relied on statistical and anomaly-based methods (Nugent 2022; Bolton and Hand 2002), but these approaches often fall short with today's complex, high-dimensional fraud patterns. Modern solutions increasingly turn to advanced machine learning to address these challenges and bolster user trust. Network analysis (Jamshidi and Hashemi 2012) revealed how connections between entities can expose fraud, though it often relies on static data and misses evolving patterns. Customer profiling, as in (Carminati et al. 2015) and (Li et al. 2024a), clusters users by behavior to spot anomalies, but can falter when fraudsters imitate legitimate activity.

The rise of big data popularized data mining for fraud detection (Kirkos, Spathis, and Manolopoulos 2007; Ravisankar et al. 2011; Wang et al. 2023), though these methods often rely on heavy feature engineering and struggle with real-time adaptability. Machine learning advanced detection with models like decision trees, random forests, and XGBoost (Carminati et al. 2015; Shimin et al. 2020), while (Rao et al. 2020) and (Nugent 2022) tackled explainability and privacy. Still, many ML models remain opaque and falter on imbalanced data. Unsupervised learning, such as SimCLR-based contrastive frameworks (Li et al. 2025), aims to learn fraud representations without labels but hinges on effective augmentations and may miss subtle cues. Deep learning has further expanded the field—using CNNs, RNNs, BiLSTMs, Transformers, and hybrids (Singh et al. 2024; Feng 2025; Tian and Liu 2023; Yuan et al. 2017; Lin et al. 2024)—to capture behavioral and spatial-temporal patterns. However, these models demand large datasets, are resource-intensive, and often lack interpretability for real-time use.

Graph-based deep learning, particularly Graph Neural Networks (GNNs), has become state-of-the-art for modeling complex relational data (Wu et al. 2020). Their effectiveness in fraud detection is shown in various works (Lu et al. 2021; Yin et al. 2022; Chen et al. 2019; Lu et al. 2022; Chen et al. 2024; Jangra et al. 2023; Shao et al. 2025; Deng, Bi, and Xiao 2024; Lin et al. 2024; Feng 2025; Tian and Liu 2023; Dubey, Dubey, and Bokoro 2025; Saxena et al. 2024). GNNs support real-time Lambda architectures (Lu et al. 2021) and models like SCN_GNN leverage node and topology features (Chen et al. 2024). In e-commerce, scalable frameworks such as InfDetect (Chen et al. 2019) and GNN-EADD (Shao et al. 2025) have been proposed. Transformer-based models (Feng 2025; Tian and Liu 2023; Dubey, Dubey, and Bokoro 2025) enhance accuracy via attention across large-scale transactions. Recent work includes RAGFormer (Li et al. 2024b), which employs semantic modeling and unsupervised contrastive learning. Nonetheless, these methods often rely on clean, well-structured graph data, which isn't always available.

While most research focuses on fraud in credit cards, banking, or financial transactions, Return to Origin (RTO) abuse in e-commerce remains underexplored. (Jangra et al. 2023) addresses RTO detection via social signals like re-identification and identity duplication but overlooks behavioral and contextual factors. We extend this by incorporating order history, purchasing behavior (Carminati et al. 2015), zonal data, product-level signals, and social links to capture a fuller profile of RTO abuse. For instance, some regions show consistently high RTO rates due to logistics gaps, while certain product categories face recurrent misuse. We also leverage graph-based models (Lu et al. 2021; Shao et al. 2025) and transformer-based fraud detectors (Feng 2025; Dubey, Dubey, and Bokoro 2025; Yuan et al. 2017) to capture sequential and network-wide patterns. This multimodal architecture predicts not just who commits RTO, but why and how, enabling a unified and explainable approach.

3 Methodology

We adopt a unified architecture that integrates structural information from graph-based interactions with user behavior features to perform downstream classification for RTO detection. The proposed system consists of two principal components:

1. **GNN Module:** This module encodes relational structures between entities—primarily users and suppliers—into dense embeddings. By leveraging heterogeneous connections such as identity overlaps and transactional behaviors, the GNN captures latent associations that may not be evident through direct interactions alone.
2. **Transformer-based Sequence Encoder:** Building upon the structural embeddings produced by the GNN, this component incorporates a wide range of features including numerical, categorical, and sequential behavioral attributes. It models user history and context using temporal positional encoding mechanisms designed to highlight patterns relevant for detecting RTO.

3.1 Graph Construction

Let $G = (V, E)$ denote the heterogeneous graph constructed over a fixed historical transaction window. The node set V comprises of two distinct entity types: users and suppliers. Formally,

$$V = V_U \cup V_S \tag{1}$$

where V_U represents the set of user nodes and V_S denotes the set of supplier nodes. Each node $v \in V$ is associated with a feature vector $\boldsymbol{x}_v \in \mathbb{R}^{d_v}$, where d_v denotes the raw feature dimensionality specific to the node type.

For each user node $u \in V_U$ ($|V_U| = n_U$), the feature vector $\mathbf{x}_u \in \mathbb{R}^{150}$ encodes behavioral and transactional attributes—such as ordering frequency, return and cancellation rates, delivery success history, and region-specific behavioral signals. Likewise, each supplier node $s \in V_S$ ($|V_S| = n_S$) is associated with a vector $\mathbf{x}_s \in \mathbb{R}^{75}$, capturing seller-level metrics including fulfillment efficiency, dispute incidence, and customer interaction patterns—essential for differentiating between trustworthy and high-risk suppliers. The selection of both user and supplier features is domain guided.

We define three categories of edges in the heterogeneous graph: user–user (E_{UU}), supplier–supplier (E_{SS}), and user–supplier (E_{US}). Thereby, the edge set $E = E_{UU} \, U \, E_{SS} \, U \, E_{US}$. Each edge $(v_i, v_j) \in E^{(r)}$ is associated with a relation type $r \in \{UU, US, SS\}$, and is represented by an edge-specific feature vector $\mathbf{e}_{ij}^{(r)} \in \mathbb{R}^{d_r}$ where d_r denotes the edge feature dimension specific to the relation type.

User–user edges are added between user nodes that share at least one identity-related attribute, such as bank account number, UPI[1] ID, email address, phone number, device ID, or Google Advertising ID (GAID). These edges model identity overlaps that often indicate account duplication. The corresponding edge feature vector encodes the count of shared identifiers.

Supplier–supplier edges are defined to capture potential collusion between sellers. An edge is instantiated between two supplier nodes if any of the following conditions are satisfied: (i) they share identifiers such as tax registration ID, bank account, phone number, or device ID; (ii) they offer visually similar catalogs, measured via cosine similarity of BEiT (Bao et al. 2022) image embeddings of catalog images exceeding a threshold; or (iii) they exhibit significant overlap in their transacting user base. The supplier–supplier edge feature vector encodes the count or magnitude of these signals.

User–supplier edges are formed to represent transactional and identity-based interactions between users and sellers. An undirected edge from user u to supplier s is added if the two share identifying attributes (e.g., phone number, device ID), or if the user exhibits significant engagement or adverse outcomes with that supplier. The user–supplier edge feature vector encodes the count or magnitude of these signals.

A sample representation of the described heterogeneous graph has been depicted in Figure 1.

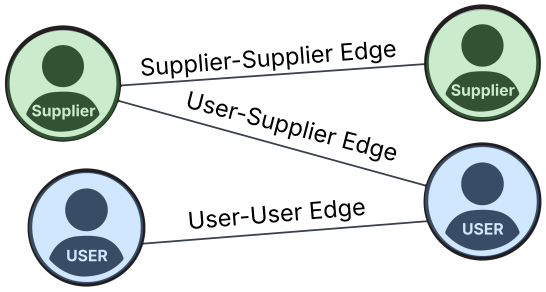

Figure 1: Sample representation of heterogeneous graph

3.2 GNN Training

We train a heterogeneous GNN to learn structural representations of users and suppliers for downstream RTO detection. The model operates over a multi-relational graph derived from historical transactional data, constructed using a holdout set of user nodes with known RTO labels from a 15–20 day observation window. These labeled nodes, referred to as core nodes, serve as the supervision anchor, while the surrounding graph provides structural context during message passing.

To construct the training graph for each of the observation day, we begin with the core nodes and iteratively expand the graph by traversing neighbor relationships up to six hops. This yields a set of training subgraphs G_{train}. The six-hop expansion depth is selected to match the model's receptive field: with three Graph Convolution (GC) layers and two-hop neighbor sampling per layer, information can flow across paths of up to six edges. While the core nodes receive direct supervision, the support nodes contribute by propagating neighborhood signals during training. Prior to training, the node feature vectors x_v are bucketized into 20 equal quantile bins and mapped to 16-dimensional embeddings using trainable lookup tables. The concatenated embeddings form the initial input representation, for each node. These inputs are projected to a shared 32-dimensional latent space, $\tilde{x}_v$, and then processed by a stack of three GC layers. Each layer $\text{GNN}^{(l,r)}$ (where l:layer index; r: relation-type) performs relation-specific message passing through, with their outputs concatenated to form the input, $\tilde{x}_v^{(l)}$, to the next layer. After the final GC layer, a multi-layer perceptron (MLP) head projects the embeddings to prediction scores giving a dense structural representation h_v of dimension d_{gnn}. The complete architecture is illustrated in Figure 2. This architecture can be formalized as:

$$\tilde{x}_{v_i}^{(l,r)} = \text{GNN}^{(l,r)} \left(x_{v_i}^{(l-1)}, \left\{ \tilde{x}_{v_j}^{(l-1)} : e_{ij}^{(r)} \in E^{(r)} \right\} \right) \quad (2)$$

$$\tilde{x}_{v_i}^{(l)} = \begin{cases} \left[\tilde{x}_{v_i}^{(l,UU)}, \; \tilde{x}_{v_i}^{(l,US)} \right] & \text{if } v_i \in V_U \\ \left[\tilde{x}_{v_i}^{(l,SS)}, \; \tilde{x}_{v_i}^{(l,SU)} \right] & \text{if } v_i \in V_S \end{cases} \quad (3)$$

$$h_{v_i} = \left[\tilde{x}_{v_i}^{(3)} \; ; \; \text{MLP}(\tilde{x}_{v_i}^{(3)}) \right] \in \mathbb{R}^{d_{gnn}} \quad (4)$$

The model is trained to minimize binary cross-entropy loss over the labeled core nodes. Let $\hat{y}_u \in [0, 1]$ denote the predicted RTO probability for user u, and $y_u \in \{0, 1\}$ be the ground-truth label. The loss is given by:

$$L_{GNN} = -\frac{1}{|V_{\text{core}}|} \sum_{u \in V_{\text{core}}} [y_u \log(\hat{y}_u) + (1 - y_u) \log(1 - \hat{y}_u)] \quad (5)$$

To scale training to large graphs with hundreds of millions of nodes and billions of edges, we adopt mini-batch neighborhood sampling. Each training batch consists of 2048 core nodes sampled from G_{train}. For each relation type, up to 10 neighbors are drawn, maintaining relational diversity during aggregation. Inference is performed daily using a freshly constructed graph built from the preceding six months of transactional activity, enabling dynamic updates of embeddings for all active users and suppliers. To address the compounding effect of severe class imbalance (Ma et al. 2025; Liu et al. 2025) along the GNN passes, we implement a historical (past 6 months) RTO-rate-weighted neighbor sampling strategy. Instead of uniform node sampling, we assign higher sampling probabilities to nodes having higher historical RTOs. The sampling weight $\boldsymbol{sw}_{e_{ij}}$ is defined as:

[1]https://www.npci.org.in/what-we-do/upi/product-overview

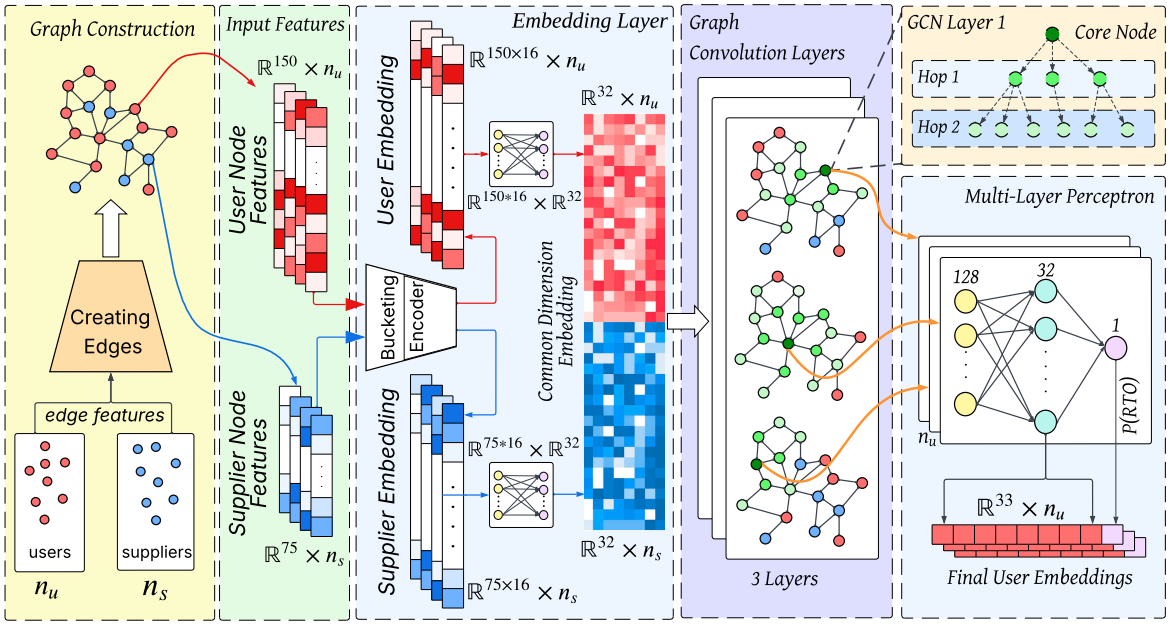

Figure 2: GNN Architecture

$$\boldsymbol{w}_e = (\omega_{v_i} + \epsilon)\,(\omega_{v_j} + \epsilon)\,, \omega_v = \frac{\text{no.}(RTO_v^{(6\text{months})})}{\text{no.}(order_v^{(6\text{months})})} \quad (6)$$

where ω_v denotes the observed RTO rate of node v over the prior six months, and ϵ is a small constant to ensure numerical stability. This weighted scheme increases the likelihood of sampling edges from RTO prone neighborhoods, thereby enhancing the model's ability to learn minority-class representations. The implementation, depicted in Figure 3, is realized using edge-level sampling weights within the PyTorch Geometric (Fey and Lenssen 2019) framework.

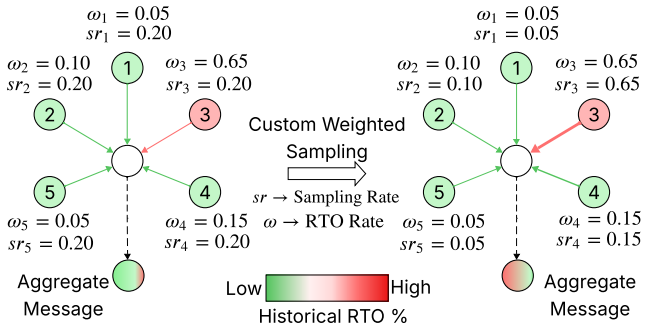

Figure 3: Weighted Neighbour Sampling based on RTO%

3.3 Transformer

The Transformer module requires a fixed-length, dense input sequence capable of representing heterogeneous data types, including structural embeddings, static contextual features, and temporal behavioral history. We construct a unified input representation that embeds graph-derived user signals, numerical and categorical features, and order history into a common latent space of dimensionality d_{model}. This section outlines the embedding process for each modality and the composition of the final input sequence.

Graph Embedding: Each user node u is associated with a dense structural representation $\boldsymbol{h}_u$ (4), generated by the upstream GNN. This embedding encodes the user's topological and relational context within the user–supplier interaction graph. To align with the Transformer input dimensionality, we apply a linear transformation:

$$\gamma_u = W_{gnn}\boldsymbol{h}_u + \boldsymbol{B}_{gnn}, \quad \gamma_u \in \mathbb{R}^{d_{\text{model}}} \quad (7)$$

The resulting vector γ_u serves as a compact representation of the user's structural identity.

Numerical Features: We select a set of domain-guided static numerical features $\mathcal{F}_{num}$ ($|\mathcal{F}_{num}| = f_n$). Each numerical feature $z_i \in \mathbb{R}$ is first quantile-binned and then encoded using a Piecewise Linear Encoding (PLE) (Gorishniy, Rubachev, and Babenko 2023) scheme. Given $T+1$ bin boundaries $\{b_0, b_1, \ldots, b_T\}$, the feature is encoded as:

$$\text{PLE}(z_i) = [\kappa_1, \ldots, \kappa_T], \quad \kappa_t = \begin{cases} 0, & z_i < b_{t-1} \\ 1, & z_i \geq b_t \\ \frac{z_i - b_{t-1}}{b_t - b_{t-1}}, & \text{otherwise} \end{cases} \quad (8)$$

This sparse representation is projected into the latent space using a linear transformation:

$$\boldsymbol{\gamma}_{z_i} = W_{\text{num}}^{(i)} \cdot \text{PLE}(z_i) + \boldsymbol{B}_{\text{num}}^{(i)}, \quad \boldsymbol{\gamma}_{z_i} \in \mathbb{R}^{d_{\text{model}}} \quad (9)$$

Stacking across all numerical features yields the matrix:

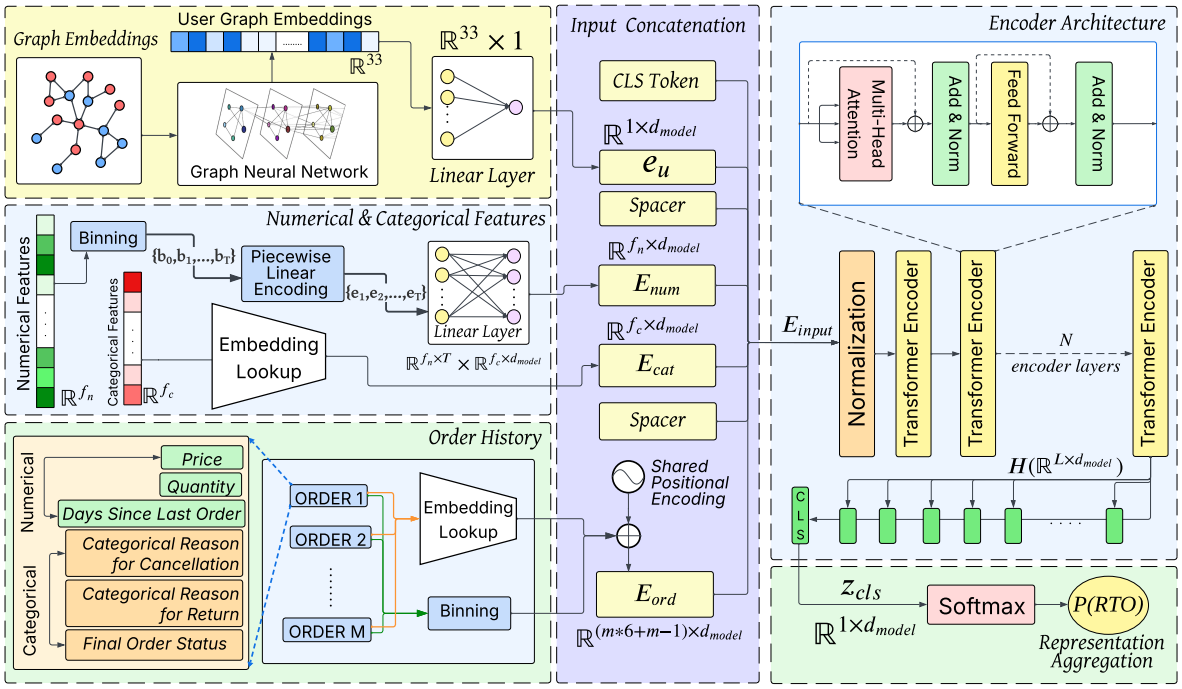

Figure 4: Transformer Model Architecture

$$\mathbf{\Gamma}_{\text{num}} = \left[\boldsymbol{\gamma}_{z_1}; \ldots; \boldsymbol{\gamma}_{z_{f_n}}\right] \in \mathbb{R}^{f_n \times d_{\text{model}}} \tag{10}$$

Categorical Features: We select a set of domain-guided static categorical features $\mathcal{F}_{cat}$ ($|\mathcal{F}_{cat}| = f_c$). $\mathcal{F}_{cat}$ includes fields such as user type (individual/re-seller), user preferred search method, user preferred language, etc. Each categorical variable $c_j \in \{1, \ldots, C_j\}, C_j \in \mathbb{Z}$ is embedded via a dedicated lookup table giving $\boldsymbol{\gamma}_{c_j}$:

$$\boldsymbol{\gamma}_{c_j} = \text{Embed}^{(j)}_{(cat)}(c_j), \quad \boldsymbol{\gamma}_{c_j} \in \mathbb{R}^{d_{\text{model}}} \tag{11}$$

The full categorical embedding matrix is then:

$$\mathbf{\Gamma}_{\text{cat}} = \left[\boldsymbol{\gamma}_{\gamma_1}; \ldots; \boldsymbol{\gamma}_{c_{f_c}}\right] \in \mathbb{R}^{f_c \times d_{\text{model}}} \tag{12}$$

Temporal Modeling of Order Behavior: Temporal dynamics play a critical role in modeling RTO, where behavioral shifts often emerge across a sequence of user transactions. To capture these patterns, we develop an encoding scheme that integrates both the semantic structure of individual orders and their temporal ordering within a user's recent history. For each user, we extract up to $m \leq 50$ of their most recent orders, preserving their original chronological order. Each order $o^{(i)}$ is represented as a six-dimensional feature tuple:

$$o^{(i)} = \left[of_1^{(i)}, of_2^{(i)}, \ldots, of_6^{(i)}\right]. \tag{13}$$

These features include the final status of the order (e.g., delivered or cancelled), the total price, item quantity, the number of days since the order was placed, a categorical cancellation reason, and a categorical return reason indicating the justification for item rejection during return. Each feature $of_j^{(i)}$ is embedded independently into a shared latent space of dimension d_{model}. Categorical features are mapped via learned embedding tables. Numerical features are first quantile-binned, then encoded using PLE, and finally projected with a linear transformation shared across orders:

$$\boldsymbol{\gamma}^{(i)}_{(ord,j)} = \begin{cases} \text{Embed}^{(j)}_{(ord)}(of_j^{(i)}), & \text{if categorical} \\ \mathbf{W}^{(i)}_{(\text{ord, num})} \cdot \text{PLE}(of_j^{(i)}) + \boldsymbol{B}^{(j)}_{(\text{ord, num})}, & \text{if numerical} \end{cases} \tag{14}$$

To preserve temporal structure, we apply a shared positional encoding across all fields of a given order. Let i denote the reverse-chronological index of order $o^{(i)}$. The positional encoding, $PE(i)$, is added uniformly to all six feature embeddings of the corresponding order:

$$\tilde{\boldsymbol{\gamma}}^{(i)}_{(ord,j)} = \boldsymbol{\gamma}^{(i)}_{(ord,j)} + \text{PE}(p_i). \tag{15}$$

This shared positional anchor ensures that all features within a transaction are temporally aligned, encouraging the encoder to treat them as a single behavioral snapshot. To form the final sequence, the six position-aware embeddings from each order are concatenated in temporal order. Spacer tokens `[S]` are inserted between orders to mark explicit transaction boundaries:

$$\begin{aligned} \mathbf{\Gamma}_{\text{ord}} = \Big[&\tilde{\boldsymbol{\gamma}}^{(1)}_{(ord,1)}, \ldots, \tilde{\boldsymbol{\gamma}}^{(1)}_{(ord,6)}, \texttt{[S]}, \ldots, \texttt{[S]}, \\ &\tilde{\boldsymbol{\gamma}}^{(m)}_{(ord,1)}, \ldots, \tilde{\boldsymbol{\gamma}}^{(m)}_{(ord,6)} \Big] \in \mathbb{R}^{\left(m \cdot 6 + m - 1\right) \times d_{\text{model}}}. \end{aligned} \tag{16}$$

Final Input Sequence: To enable multimodal learning in a unified attention mechanism, we concatenate all embeddings with special tokens:

$$\mathbf{\Gamma}_{\text{input}} = \Big[\texttt{[CLS]}; \boldsymbol{\gamma}_u; \texttt{[S]}; \mathbf{\Gamma}_{\text{num}}; \mathbf{\Gamma}_{\text{cat}}; \texttt{[S]}; \mathbf{\Gamma}_{\text{ord}}; \texttt{[S]} \Big] \tag{17}$$

Each token in $\mathbf{\Gamma}_{\text{input}}$ is further augmented with positional and modality-type encodings (described in the next section) to disambiguate field roles and temporal structure. The final sequence is then passed to the Transformer encoder for contextual representation learning and classification.

Sequence Aggregation and Classification The Transformer encoder produces contextualized representations for each input token in the sequence. Let $\boldsymbol{H} \in \mathbb{R}^{L \times d_{\text{model}}}$ denote the output matrix, where L is the total number of tokens and $\boldsymbol{H}_i \in \mathbb{R}^{d_{\text{model}}}$ corresponds to the representation of token i. To obtain a fixed-size representation for downstream classification, we evaluate multiple aggregation strategies:

CLS Token Representation: A special `[CLS]` token is prepended to the input. Its output representation $\boldsymbol{z} = \boldsymbol{H}_1$ is used as the summary vector. This method showed the best empirical performance and is adopted in the final model.

Average Pooling: The representation is computed as the mean of all token embeddings (excluding special tokens):

$$\boldsymbol{z} = \frac{1}{|M|} \sum_{i \in M} \boldsymbol{H}_i \tag{18}$$

where M is the set of valid tokens.

Attention Pooling: A learnable vector $\boldsymbol{w} \in \mathbb{R}^{d_{\text{model}}}$ is used to compute attention weights over tokens:

$$\alpha_i = \frac{\exp(\boldsymbol{W}_{att}^{\top} \tanh(\boldsymbol{H}_i))}{\sum_{j \in M} \exp(\boldsymbol{W}_{att}^{\top} \tanh(\boldsymbol{H}_j))}, \quad \boldsymbol{z} = \sum_{i=1}^{M} \alpha_i \boldsymbol{H}_i \tag{19}$$

In all cases, the final representation $\boldsymbol{z}$ is passed through a linear classifier to produce logits:

$$\hat{\boldsymbol{y}} = \text{softmax}(W_{\text{cls}} \boldsymbol{z} + \boldsymbol{b}_{\text{cls}}) \tag{20}$$

We model the problem as binary classification, where the predicted RTO probability is $\hat{y}_1 = P(y = 1)$ and $y \in \{1, 0\}$ is groundtruth label. To address the class imbalance inherent in RTO prediction, we use a weighted binary cross-entropy loss:

$$\mathcal{L}_{\text{wce}} = \frac{-1}{N} \sum_{i=1}^{N} (\mu_1 \cdot y \log(\hat{y}_i) + \mu_0 \cdot (1 - y) \log(1 - \hat{y}_i)) \tag{21}$$

where class weights μ_0 and μ_1 are inversely proportional to label frequencies in the training set. The overall model architecture, including the aggregation and classification layers, is illustrated in Figure 4.

4 Experiments

We structure our experiments around the following research questions, each addressing a key challenge in RTO detection:

- **RQ1:** Is our model more effective than other baselines?
- **RQ2:** How can complex dependencies across users, and suppliers be captured to detect community-level abuse?
- **RQ3:** How can we identify hidden RTO risks in users who appear clean but are structurally linked to high-risk entities?
- **RQ4:** How can we predict RTO risk for users with limited or sparse behavioral history?
- **RQ5:** How can temporal volatility in user behavior be captured to improve RTO prediction accuracy?

We used two key metrics to evaluate each experiment: **AUCPR** and **Precision**.

4.1 Experimental Setup

Dataset We use a large-scale e-commerce dataset with millions of COD transactions, including user interactions, order histories, and delivery outcomes, with strict user-level splits to prevent leakage (Table 1). All baselines are evaluated on the full test set for RQ1, while RQ2–RQ5 use curated 5,000-user cohorts labeled with domain-expert input.

Split	#Orders	#RTO	#Users
Train	10,650,370	2,248,707	6,510,493
Test	1,004,484	211,172	678,708
Val	877,939	185,487	579,896

Table 1: Dataset Statistics

4.2 Performance Comparison (RQ1)

We benchmark TRUST against a diverse set of models widely used in e-commerce for RTO detection on our dataset. As shown in Table 2, TRUST outperforms all baselines ranging from tabular, deep learning, transformer-based, graph-based, and unified model achieving an AUCPR of 0.47 and Precision of 68%. These results underscore the advantage of TRUST's graph-transformer architecture in effectively identifying fraudulent RTO behavior.

Architecture	Model	AUCPR	Precision (%)
Tabular	XGBoost	0.34	58.7
	(Jangra et al. 2023)	0.36	60.1
DNN	(Yuan et al. 2017)	0.3	55.3
Transformer	(Li et al. 2025)	0.38	61.8
	(Feng 2025)	0.37	60.9
	(Deng, Bi, and Xiao 2024)	0.36	59.6
Graph	(Yin et al. 2022)	0.26	52.7
	(Rao et al. 2020)	0.24	50.8
	(Lu et al. 2022)	0.24	51
	(Shao et al. 2025)	0.3	55.5
	(Hooi et al. 2016)	0.27	53.2
	(Chen et al. 2024)	0.31	56.3
Unified	(Lin et al. 2024)	0.45	66.9
	(Li et al. 2024b)	0.28	53.8
	TRUST	**0.47**	**68.3**
	(Singh et al. 2024)	0.44	65.7
	(Tian and Liu 2023)	0.39	62.5
	(Dubey, Dubey, and Bokoro 2025)	0.31	56.4

Table 2: Comparison of RTO Detection Models

4.3 Capturing multi-entity dependencies (RQ2)

To evaluate the role of multi-entity dependencies in RTO detection, we constructed a cohort of 5,000 users with non-anomolous previous history but structural links to other users or suppliers via shared attributes (e.g., address, device, phone). We compared two TRUST variants: one with graph-based entity embeddings and one using only behavioral features, trained under identical settings. As shown in Table 3 and also in Figure 5, the graph-augmented model uncovers dense clusters of interconnected users and suppliers with elevated RTO risk, while the behavior-only variant misses these patterns—highlighting graph reasoning as necessary to surface fraud rings and collusive entities undetected by behavioral models.

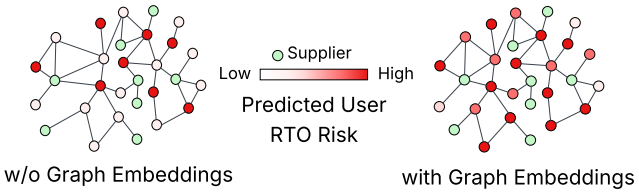

Figure 5: RQ2 Analysis

RQ	Experiment	AUCPR	Precision (%)
2	**TRUST**	**0.46**	**68.1**
	-w/o GE	0.43	66.4
3	**TRUST**	**0.36**	**63.4**
	-w/o GE	0.27	58.9
4	**TRUST**	**0.34**	**62.5**
	-w/o GE	0.28	59.3
5	Graph-Only	0.24	57.6
	TRUST	**0.47**	**68.4**

Table 3: Performance for each experiment. Here, *-w/o GE* denotes the variant trained without Graph Embeddings.

4.4 Identifying hidden RTO risks (RQ3)

Similar to RQ2, we also constructed a cohort of 5,000 users with no prior RTOs or cancellations but structurally linked to high-risk entities via shared addresses, devices, or suppliers. We compared two TRUST variants: one with only behavioral features and another with graph-based embeddings, both trained identically and calibrated with Platt scaling. As shown in Table 3, the graph-enhanced model achieves higher AUCPR and precision, with right-skewed probability distributions that flag high-risk users despite clean histories—patterns missed by the Transformer-only baseline. This underscores the importance of structural context and multi-hop reasoning in surfacing latent RTO risk.

4.5 RTO risk with limited order history (RQ4)

Users with limited order history pose a challenge for behavior-based RTO prediction models due to sparse data.

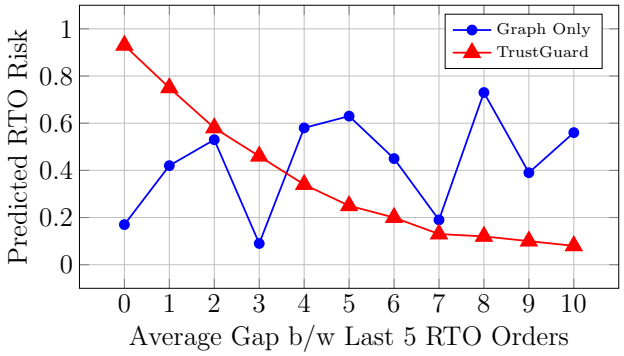

Figure 6: RQ5 Analysis

To evaluate TRUST's robustness in cold-start scenarios, we use a test cohort of 5,000 users with $\leq$ 4 orders—representing typical new or infrequent users. We compare two model variants: one using only behavioral features, and another augmented with graph-based embeddings. Both are trained identically to isolate the impact of structural signals. As shown in Table 3, the graph-enhanced model consistently outperforms the baseline, effectively compensating for limited history by leveraging connections to high-risk entities via message passing.

4.6 Temporal volatility in user behavior (RQ5)

We evaluate TRUST's ability to capture temporal volatility by testing whether users with closely clustered RTOs receive higher risk scores than those with evenly spaced events. We construct a cohort of 5,000 users, each with five RTOs in their last 50 orders, with the most recent order being an RTO. By comparing users based on their average inter-RTO gap, we isolate the effect of temporal clustering. TRUST, which incorporates temporal modeling, assigns significantly higher risk to users with recent, frequent RTOs (Figure 6), while the graph-only baseline shows weak or no correlation. Results are summarized in Table 3.

5 Real World Deployment

5.1 System Architecture

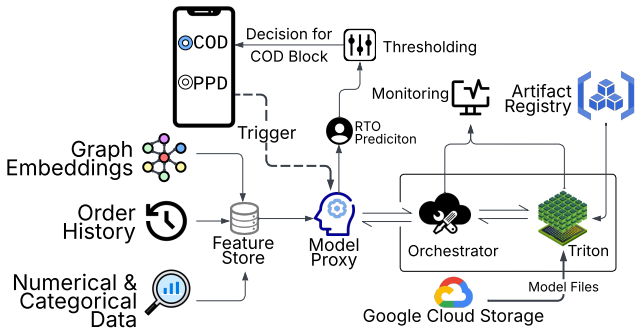

Figure 7: End-to-End Inference Workflow

At payment page, RTO risk is scored via a low latency model serving(model proxy) using daily refreshed features from Feature store(FS). The Model Proxy handles inputs, routes inference, and enforces threshold rules for high-risk COD, while enabling versioning and experiments. Models are stored in GCS, versioned via Artifact Registry, and served through an orchestrator managing batching, load balancing, telemetry, and A/B tests. Latency, errors, throughput, and GPU usage are tracked in Grafana with anomaly alerts for fast recovery.

5.2 Inference Optimization

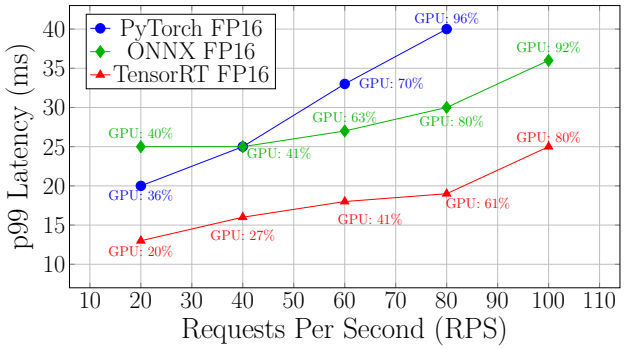

Figure 8: Latency vs. Throughput across Serving Backends

Real time models are hosted in TensorRT, doubling throughput at similar latency under 80% GPU load. Inference runs on NVIDIA Triton with TensorRT optimized Transformer models (FP16) with AMP. Model proxy has p99 latency of <50 ms with entity level FS latency of <10 ms & Model serving latency of <30 ms (Figure 8).

5.3 Operational Considerations

To ensure robustness in live environments, the following operational strategies were incorporated:

- **Threshold Calibration:** Decision thresholds for COD blocking are periodically recalibrated to optimize precision–recall tradeoffs in alignment with evolving business objectives.
- **Scalable Infrastructure:** GPU-backed inference nodes auto-scale based on seasonal traffic surges (e.g., during festive sales), with scale-down policies to minimize cost.
- **Model Refresh and Rollout:** New models are trained on a rolling window of behavioral data, validated offline, and deployed through canary and A/B testing. Only upon statistically significant improvement are model versions scaled in production.
- **Observability:** System-wide observability is maintained through Grafana dashboards tracking request latencies, backend health, error rate, failure counts and GPU utilization.

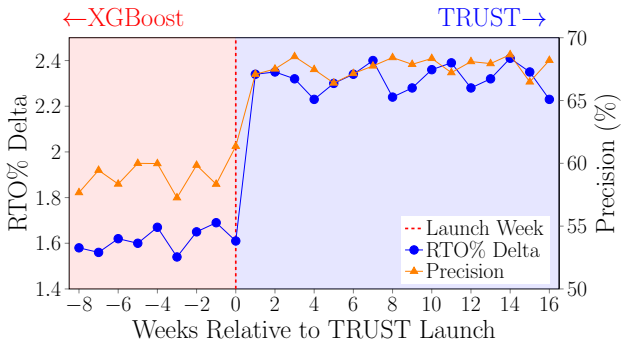

Figure 9: Post Deployment Performance Comparison

5.4 Post Launch Performance

We evaluated **TRUST** in production via an **A/B test** against the previously deployed **XGBoost model** on a high-throughput, pan-India e-commerce platform. The experiment began with a **5% test** and **5% control** group, followed by progressive scaling to **20%**, **50%**, and finally **95%** of traffic. Figure 9 summarizes the **24-week timeline**—the first 8 weeks under XGBoost and the next 16 under TRUST. The following metrics were monitored weekly:

- **RTO% Delta:** Difference in Return-to-Origin rates between test and control.
- **Precision:** Evaluated at fixed recall, corresponding to a constant daily COD block rate.
- **Pre-Order:** Total carts/users blocked; first-time COD blocks; precision/recall; app adoption; leakage due to timeouts (>200 ms) or rate-limiters.
- **Post-Order:** Blocked carts converted to prepaid; cancellations; precision/recall; impacted orders; holdout/non-holdout COD RTO rates.

TRUST reduced RTO significantly and improved precision by **∼9.6% at fixed recall**, yielding an estimated **∼5pp reduction in operational expenses**. Gains remained stable over 16 weeks despite traffic and product fluctuations, demonstrating scalability across millions of daily orders.

Retraining Frequency and Cost: Model metrics (RTOΔ, precision, recall) were tracked weekly. Retraining was triggered only on significant trend shifts, occurring roughly every **6–8 months** at a cost of **$150–$200** per cycle, balancing performance stability with operational efficiency.

6 Conclusion & Future Work

We present TRUST, a real-time RTO detection system that combines graph neural networks and Transformer-based sequence models to combat RTO abuse in e-commerce. By jointly modeling structural and behavioral signals, TRUST identifies complex RTO patterns such as identity overlap, collusion, and repeat abuse, achieving a ∼9.6% precision gain over prior methods with ∼25 ms inference latency at scale. Future work will expand the graph schema and incorporate causal inference to enhance interpretability and robustness against adversarial behavior.

References

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2022. BEiT: BERT Pre-Training of Image Transformers. arXiv:2106.08254.

Bolton, R. J.; and Hand, D. J. 2002. Statistical fraud detection: A review. *Statistical science*, 17(3): 235–255.

Carminati, M.; Caron, R.; Maggi, F.; Epifani, I.; and Zanero, S. 2015. BankSealer: A decision support system for online banking fraud analysis and investigation. *computers & security*, 53: 175–186.

Chen, C.; Liang, C.; Lin, J.; Wang, L.; Liu, Z.; Yang, X.; Zhou, J.; Shuang, Y.; and Qi, Y. 2019. InfDetect: A large scale graph-based fraud detection system for E-commerce insurance. In *2019 IEEE International Conference on Big Data (Big Data)*, 1765–1773. IEEE.

Chen, J.; Chen, Q.; Jiang, F.; Guo, X.; Sha, K.; and Wang, Y. 2024. SCN_GNN: A GNN-based fraud detection algorithm combining strong node and graph topology information. *Expert Systems with Applications*, 237: 121643.

Dazeinfo. 2024. Over 25% of COD Orders Fail: A Major Dent in India's E-commerce Business. https://dazeinfo.com/2024/06/05/over-25-of-cod-orders-fail-a-major-dent-in-indias-e-commerce-business/.

Deng, T.; Bi, S.; and Xiao, J. 2024. Transformer-based financial fraud detection with cloud-optimized real-time streaming. In *Proceedings of the 2024 5th International Conference on Big Data Economy and Information Management*, 702–707.

Dubey, P.; Dubey, P.; and Bokoro, P. N. 2025. A Unified Transformer–BDI Architecture for Financial Fraud Detection: Distributed Knowledge Transfer Across Diverse Datasets. *Forecasting*, 7(2): 31.

Feng, P. 2025. Hybrid BiLSTM-Transformer Model for Identifying Fraudulent Transactions in Financial Systems. *Journal of Computer Science and Software Applications*, 5(3).

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

Gorishniy, Y.; Rubachev, I.; and Babenko, A. 2023. On Embeddings for Numerical Features in Tabular Deep Learning. arXiv:2203.05556.

Hooi, B.; Song, H. A.; Beutel, A.; Shah, N.; Shin, K.; and Faloutsos, C. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 895–904.

Jamshidi, S.; and Hashemi, M. R. 2012. An efficient data enrichment scheme for fraud detection using social network analysis. In *6th International Symposium on Telecommunications (IST)*, 1082–1087. IEEE.

Jangra, H.; K, A.; Saha, S.; Banerjee, S.; Chelliah, M.; and Kumaraguru, P. 2023. Social re-identification assisted rto detection for e-commerce. In *Companion Proceedings of the ACM Web Conference 2023*, 854–858.

Juniper Research. 2024. eCommerce Fraud to Exceed $107 Billion in 2029. https://www.juniperresearch.com/press/pressreleasesecommerce-fraud-to-exceed-107bn-in-2029/.

Kirkos, E.; Spathis, C.; and Manolopoulos, Y. 2007. Data mining techniques for the detection of fraudulent financial statements. *Expert systems with applications*, 32(4): 995–1003.

LexisNexis. 2024. Every Rupee Lost to Fraud in India Costs Firms Rs. 4. https://risk.lexisnexis.com/global/en/about-us/press-room/press-release/20240429-tcof-india.

Li, H.; Jiang, S.; Zhang, L.; Du, S.; Ye, G.; and Chai, H. 2024a. Fraud Detection with Binding Global and Local Relational Interaction. *arXiv e-prints*, arXiv–2402.

Li, H.; Jiang, S.; Zhang, L.; Du, S.; Ye, G.; and Chai, H. 2024b. RAGFormer: Learning Semantic Attributes and Topological Structure for Fraud Detection. *arXiv preprint arXiv:2402.17472*.

Li, X.; Peng, Y.; Sun, X.; Duan, Y.; Fang, Z.; and Tang, T. 2025. Unsupervised Detection of Fraudulent Transactions in E-commerce Using Contrastive Learning. In *2025 4th International Symposium on Computer Applications and Information Technology (ISCAIT)*, 1663–1667. IEEE.

Lin, J.; Guo, X.; Zhu, Y.; Mitchell, S.; Altman, E.; and Shun, J. 2024. FraudGT: A Simple, Effective, and Efficient Graph Transformer for Financial Fraud Detection. In *Proceedings of the 5th ACM International Conference on AI in Finance*, 292–300.

Liu, Z.; Li, Y.; Chen, N.; Wang, Q.; Hooi, B.; and He, B. 2025. A survey of imbalanced learning on graphs: Problems, techniques, and future directions. *IEEE Transactions on Knowledge and Data Engineering*.

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*, 3342–3351.

Lu, M.; Han, Z.; Zhang, Z.; Zhao, Y.; and Shan, Y. 2021. Graph neural networks in real-time fraud detection with lambda architecture. *arXiv preprint arXiv:2110.04559*.

Ma, Y.; Tian, Y.; Moniz, N.; and Chawla, N. V. 2025. Class-imbalanced learning on graphs: A survey. *ACM Computing Surveys*, 57(8): 1–16.

Nugent, D. 2022. Privacy-preserving credit card fraud detection using homomorphic encryption. *arXiv preprint arXiv:2211.06675*.

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*.

Ravisankar, P.; Ravi, V.; Rao, G. R.; and Bose, I. 2011. Detection of financial statement fraud and feature selection using data mining techniques. *Decision support systems*, 50(2): 491–500.

Saxena, S.; Ghatak, S.; Kolla, R.; Mukherjee, D.; and Chakraborty, T. 2024. Dphgnn: A dual perspective hypergraph neural networks. In *Proceedings of the 30th ACM*

SIGKDD Conference on Knowledge Discovery and Data Mining, 2548–2559.

Shao, Z.; Wang, X.; Ji, E.; Chen, S.; and Wang, J. 2025. GNN-EADD: Graph Neural Network-based E-commerce Anomaly Detection via Dual-stage Learning. *IEEE Access*.

Shimin, L.; Ke, X.; Xinye, S.; et al. 2020. An Xgboost based system for financial fraud detection. In *E3S Web of Conferences*, volume 214, 02042. EDP Sciences.

Singh, M. T.; Prasad, R. K.; Michael, G. R.; Kaphungkui, N.; and Singh, N. H. 2024. Heterogeneous Graph Auto-Encoder for CreditCard Fraud Detection. *arXiv preprint arXiv:2410.08121*.

Tian, Y.; and Liu, G. 2023. Transaction fraud detection via spatial-temporal-aware graph transformer. *arXiv preprint arXiv:2307.05121*.

Wang, Z.; Wu, Q.; Zheng, B.; Wang, J.; Huang, K.; and Shi, Y. 2023. Sequence as genes: an user behavior modeling framework for fraud transaction detection in e-commerce. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5194–5203.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Yin, H.; Zhang, Z.; Wang, Z.; Özyurt, Y.; Liang, W.; Dong, W.; Zhao, Y.; and Shan, Y. 2022. Behavioral graph fraud detection in E-commerce. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, 1–8. IEEE.

Yuan, S.; Wu, X.; Li, J.; and Lu, A. 2017. Spectrum-based deep neural networks for fraud detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2419–2422.