Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Jacobi-Enhanced Dynamic Hypergraph Learning for Graph-Based Fraud Detection

Danyang Li
Beijing Police College
Jie Shen
Guangxi Police college
XiangBeng Yang
Guangxi Police college
Zhenkai Qin
qinzhenkai@gxjcxy.edu.cn

Guangxi Police college

Article

Keywords: Fraud detection, Dynamic hypergraph neural network, KAN-like classification, Graph neural network, Financial fraud

Posted Date: September 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10577848/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Scientific Reports on October 29th, 2026. See the published version at https://doi.org/10.1038/s41598-026-76204-6.

Jacobi-Enhanced Dynamic Hypergraph Learning for Graph-Based Fraud Detection

Danyang Li[1], Jie Shen[2], XiangBeng Yang[2], Zhenkai Qin[2*]

[1]Department of Public Security Management, Beijing Police College, Beijing, China.

[2]School of Information Technology, Guangxi Police College, Nanning, China.

*Corresponding author(s). E-mail(s): qinzhenkai@gxjcxy.edu.cn;

Abstract

Financial and e-commerce fraud detection remains challenging because fraudulent entities may exploit structural camouflage and coordinated activities. We propose JK-DHGNN, a dual-view framework that combines observed pairwise relations with learned higher-order affiliations. A Top-$\boldsymbol{K}$ dynamic hypergraph generator suppresses low-confidence assignments, a Jacobi spectral filter aggregates multi-order graph signals, and a Jacobi polynomial-based KAN-like classifier provides adaptive nonlinear prediction. Under stratified random splits, JK-DHGNN achieves the highest mean AUC and AP on YelpChi and Amazon, while remaining above several conventional and hypergraph baselines on Elliptic. Ablation analyses reveal dataset-dependent component behavior: Jacobi filtering and auxiliary supervision are most beneficial on the review graphs, whereas dynamic hypergraph learning is critical on the sparse Elliptic transaction graph. Random edge perturbation and a transductive time-aware case study further characterize the model's stability and limitations. These results support dynamic high-order structure learning for fraud detection while highlighting the need for dataset-adaptive integration of spectral and classifier modules.

Keywords: Fraud detection, Dynamic hypergraph neural network, KAN-like classification, Graph neural network, Financial fraud

1 Introduction

Financial and e-commerce fraud—including credit card fraud, money laundering, and malicious review manipulation—poses a persistent threat to digital economies and decentralized financial systems [1]. Such misconduct is rarely independent: accounts, transactions, users, and products form relational patterns that can reveal coordinated or concealed behavior. Graph neural networks (GNNs) have therefore become an important paradigm for fraud detection because they jointly exploit entity attributes and graph structure [2]. Their effectiveness, however, depends on the informativeness of the neighborhoods over which messages are propagated. Fraudsters can undermine this premise by coordinating in groups and establishing benign-looking interactions that obscure suspicious behavior. These strategies introduce heterophilous or misleading connections and reduce the reliability of conventional neighborhood aggregation [3].

Despite substantial progress, three limitations remain particularly relevant to graph-based fraud detection. **(1) Incomplete representation of higher-order coordination:** Most GNNs propagate information over pairwise edges and can represent group-level behavior only indirectly through multiple propagation steps. Pairwise graphs may therefore fail to express explicitly that several entities participate in a common fraudulent activity [4]. Furthermore, when the observed topology is incomplete or unavailable, heuristic constructions such as k-nearest-neighbor graphs can introduce spurious relations. **(2) Sensitivity to heterophily and over-smoothing:** Fraudulent entities may connect to benign entities as a form of camouflage, violating the homophily assumption implicit in many aggregation schemes. Because conventional message passing often has a low-pass effect, repeated propagation can homogenize node representations and attenuate high-frequency components that may carry anomaly-related information [5, 6]. **(3) Challenging decision-boundary estimation:** Fraudulent and benign nodes may remain entangled after graph propagation, and the scarcity of fraudulent samples further complicates classification. Although MLPs are expressive in principle, their fixed nodal activation functions may be inefficient for representing the irregular nonlinear mappings required in such settings [7].

Prior studies have investigated fraud-aware neighbor selection, graph structure learning, spectral filtering, and expressive nonlinear classifiers. Each direction addresses an important part of the problem, but their separate use leaves interactions between topology construction, graph-signal propagation, and final classification insufficiently examined. In particular, a learned higher-order structure may still propagate unreliable affiliations if it is dense; flexible spectral features may remain underused if the classifier cannot model a complex boundary; and a learned topology should complement, rather than indiscriminately replace, reliable observed relations. These considerations motivate a unified framework that retains observed structural evidence, learns sparse latent affiliations, extracts multi-order graph signals, and performs adaptive nonlinear classification in an end-to-end manner.

To this end, we propose **JK-DHGNN** (Jacobi-KAN Enhanced Dynamic Hypergraph Neural Network), a dual-view framework for fraud detection. Its static view converts each observed pairwise edge into a two-node hyperedge, preserving the available topology in a hypergraph-compatible form. In parallel, an **Adaptive Dynamic**

Hypergraph Generator infers latent node-to-hyperedge affiliations from intermediate representations. Row-wise Top-K sparsification retains only the most confident dynamic affiliations of each node, thereby limiting propagation through low-confidence learned assignments without discarding observed edges from the static view. A **Spectral Jacobi Graph Filter** constructs and adaptively combines multiple polynomial orders, providing a flexible frequency response rather than imposing purely low-pass aggregation. Finally, a **Jacobi-KAN layer** combines a SiLU-activated linear branch with learnable one-dimensional functions parameterized by Jacobi polynomials. We refer to this classifier as KAN-like because it follows the edge-function principle of KAN while replacing the original B-spline basis with Jacobi polynomials. The complete model jointly optimizes spectral representation learning, dynamic topology construction, dual-view propagation, and nonlinear prediction.

The main contributions of this paper are summarized as follows:

- **Dual-view structural learning:** We develop a hypergraph framework that preserves the observed pairwise topology as a static structural prior and learns a complementary dynamic view of latent higher-order affiliations. Top-K assignment restricts each node to its most confident learned hyperedge affiliations, reducing the influence of low-confidence dynamic relations.
- **Jacobi-enhanced representation and classification:** We integrate a Jacobi spectral graph filter with a Jacobi polynomial-based KAN-like classifier. The filter adaptively combines multi-order graph signals, whereas the classifier learns edge-wise nonlinear transformations for the resulting fraud representations.
- **Structure-aware empirical analysis:** We evaluate JK-DHGNN across dense review graphs and a sparse transaction graph through component ablations, random structural perturbation, a transductive time-aware case study, and computational-cost analysis. The results reveal both the effectiveness and the dataset-dependent interactions of the proposed components.

2 Related Work

Early financial anomaly detection methods relied primarily on conventional machine learning algorithms, such as random forests and support vector machines, or on rule-based expert systems. These approaches generally treated entities as independent observations and therefore overlooked relational dependencies. The emergence of graph neural networks (GNNs), including GCN [8] and GraphSAGE [9], enabled neighborhood-based representation learning and improved the use of local structural information. Nevertheless, many spatial message-passing models assume homophily and may perform poorly when fraudulent entities disguise themselves through interactions with benign nodes. Fraud-aware GNNs have consequently been developed to address heterophily, camouflage, and class imbalance. CARE-GNN [10], for example, uses reinforcement learning to select relation-specific neighbor-sampling thresholds, whereas PC-GNN [11] employs label-balanced sampling. FraudRE [12] further incorporates heterogeneous relations to distinguish the semantics of different interaction types. Despite these advances, most such models remain based on pairwise node-to-node propagation and thus represent coordinated group behavior only indirectly.

Hypergraph learning provides a natural means of representing relations among more than two entities. Zhou et al. [13] established a spectral formulation for hypergraph learning, and subsequent neural approaches extended this formulation to representation learning. HGNN [14] models high-order data correlations through hypergraph convolution, whereas HyperGCN [15] approximates hypergraph structure using graph-based propagation. However, conventional hypergraph methods remain difficult to apply directly to fraud detection. Static hypergraphs constructed from heuristics such as k-nearest neighbors can introduce spurious associations when the feature space is contaminated by adversarial or noisy observations. Dynamic hypergraph methods, including that of Jiang et al. [16], can adapt the structure during learning, but dense assignment or similarity operations may increase computational cost and propagate low-confidence relations. Our framework instead retains the observed pairwise topology in a static hypergraph-compatible view and learns a complementary dynamic view through Top-K sparse hyperedge assignment.

Feature propagation presents a related challenge. From a graph signal processing perspective, neighborhood aggregation often behaves as a low-pass filter. Although this behavior is beneficial in homophilous graphs, it can attenuate high-frequency components associated with anomalous or heterophilous nodes. Repeated propagation may further produce over-smoothing, whereby node representations become increasingly similar. Spectral GNNs address this limitation by parameterizing graph filters in the frequency domain. ChebNet [17] uses Chebyshev polynomials for localized filtering, while BernNet [18] and GPR-GNN [19] learn flexible frequency responses. Wang et al. [20] further showed that orthogonal Jacobi polynomials constitute a flexible basis for spectral graph filtering. Motivated by these findings, we use Jacobi polynomial filtering to combine multi-scale structural signals and retain anomaly-relevant frequency components.

At the classification stage, most GNN-based fraud detectors use multilayer perceptrons (MLPs). Although MLPs are universal approximators, architectures based on fixed nodal activation functions may not provide the most parameter-efficient representation of irregular decision boundaries under severe class imbalance. Kolmogorov-Arnold Networks (KANs) replace fixed nodal activations with learnable one-dimensional functions on network edges [21]. The original KAN parameterizes these functions using B-spline bases. In contrast, our KAN-like classifier uses Jacobi polynomial bases together with a SiLU-activated linear branch. This parameterization aligns the classifier with the spectral module and provides an adaptive nonlinear mapping for the final fraud representations.

Collectively, prior studies have advanced fraud-aware message passing, hypergraph learning, spectral filtering, and adaptive function approximation. These directions, however, have largely been investigated independently. The joint modeling of observed structural priors, latent higher-order affiliations, anomaly-relevant spectral components, and adaptive nonlinear classification remains underexplored. JK-DHGNN addresses this gap by integrating Top-K sparse dynamic hypergraph generation, Jacobi spectral filtering, and a Jacobi polynomial-based KAN-like classifier within a unified end-to-end framework.

3 Preliminaries and Problem Formulation

This section introduces the notation used for graphs and hypergraphs and formulates fraud detection as transductive semi-supervised node classification. Bold uppercase letters denote matrices (e.g., $\mathbf{X}$), bold lowercase letters denote vectors (e.g., $\mathbf{x}$), and italic lowercase letters denote scalars (e.g., k).

3.1 Notations and Definitions

Definition 1 (Attributed Graph). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an input graph, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of N nodes (e.g., accounts or transactions) and $\mathcal{E}$ is the edge set. The associated feature matrix is $\mathbf{X} \in \mathbb{R}^{N \times F}$, whose i-th row, $\mathbf{x}_i \in \mathbb{R}^F$, is the F-dimensional feature vector of node v_i.

Definition 2 (Hypergraph). A hypergraph generalizes a standard graph by allowing each hyperedge to connect more than two nodes. Let $\mathcal{G}_h = (\mathcal{V}, \mathcal{E}_h)$ denote a hypergraph with hyperedge set $\mathcal{E}_h = \{e_1, e_2, \ldots, e_M\}$. Each hyperedge $e_j \subseteq \mathcal{V}$ may contain an arbitrary number of nodes and can therefore represent group-level relations.

The topological structure of a hypergraph is represented by a nonnegative incidence matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{N \times M}$. Its entry $\mathbf{H}_{ij}$ is defined as

$$\mathbf{H}_{ij} = \begin{cases} w_{ij}, & \text{if node } v_i \in e_j, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $w_{ij} > 0$ denotes the participation weight of node v_i in hyperedge e_j. In JK-DHGNN, the static incidence matrix $\mathbf{H}_{static}$ is the binary special case with $w_{ij} = 1$, whereas the dynamic incidence matrix $\mathbf{H}_{dyn}$ is a sparse nonnegative weighted matrix obtained after Top-K sparsification and row-wise renormalization.

The node degree $d(v_i)$ and hyperedge degree $d(e_j)$ are obtained from the corresponding row and column sums of $\mathbf{H}$. The diagonal node- and hyperedge-degree matrices, $\mathbf{D}_v \in \mathbb{R}^{N \times N}$ and $\mathbf{D}_e \in \mathbb{R}^{M \times M}$, are defined as

$$(\mathbf{D}_v)_{ii} = \sum_{j=1}^{M} \mathbf{H}_{ij}, \quad (\mathbf{D}_e)_{jj} = \sum_{i=1}^{N} \mathbf{H}_{ij} \tag{2}$$

These matrices are used to normalize hypergraph message passing.

3.2 Problem Formulation

We consider a transductive node-classification benchmark with three pairwise-disjoint labeled node sets: the training set $\mathcal{V}_{tr}$, validation set $\mathcal{V}_{val}$, and test set $\mathcal{V}_{te}$. Their union is a subset of $\mathcal{V}$ because a dataset may also contain nodes without reference labels. The complete graph structure and node features are available during training, but the optimization objective is evaluated only on $\mathcal{V}_{tr}$. The validation labels are used exclusively for hyperparameter selection and early stopping, and the test labels are accessed only for final evaluation. In the primary experiments, the labeled benchmark

nodes are divided into these three sets using the 40%/20%/40% protocol described in Section 5.

Let $\mathbf{Y}$ denote the reference labels for the benchmark nodes, where C is the number of classes. For binary fraud detection, $C = 2$, and the scalar label $y_i \in \{0, 1\}$ indicates whether node v_i is benign ($y_i = 0$) or fraudulent ($y_i = 1$).

Objective: Given the node feature matrix $\mathbf{X}$, the training labels $\mathbf{Y}[\mathcal{V}_{tr}]$, and an initial topology represented by the static incidence matrix $\mathbf{H}_{static}$, JK-DHGNN learns a mapping $f_\Theta(\cdot)$ parameterized by Θ:

$$f_\Theta(\mathbf{X}, \mathbf{H}_{static}) = \hat{\mathbf{Y}} \tag{3}$$

The objective is to predict the labels of nodes in $\mathcal{V}_{te}$ while reducing sensitivity to heterophilous and noisy graph structure and severe class imbalance, without using validation or test labels in parameter optimization.

4 Methodology

4.1 Overall Framework

Fraudulent entities may conceal their behavior through benign-looking local interactions while simultaneously participating in coordinated group activities. These patterns challenge conventional GNNs, which generally rely on pairwise propagation and may suffer from over-smoothing in heterophilous graphs. JK-DHGNN addresses these issues by extracting multi-scale structural signals, learning latent higher-order affiliations, and providing adaptive nonlinear classification under severe class imbalance.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the input graph, where $\mathcal{V}$ contains N nodes and $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the node feature matrix. The initial topology may be observed directly from relational data or constructed from node features. To support hypergraph message passing, reciprocal and duplicated node pairs are coalesced, and each remaining non-self-loop pair $\{u, v\}$ is converted into one two-node hyperedge, yielding the static incidence matrix $\mathbf{H}_{static}$. The static view therefore retains the available pairwise structural prior, whereas the dynamic view learns latent higher-order affiliations. The spectral adjacency uses the same coalesced node pairs and represents directed relations bidirectionally before normalization.

As illustrated in Figure 1, JK-DHGNN comprises three components. **Spectral Jacobi Feature Extraction** applies orthogonal Jacobi polynomial filtering to obtain multi-scale graph representations. **Adaptive Dynamic Hypergraph Generation** infers a sparse incidence matrix $\mathbf{H}_{dyn}$ through Top-K hyperedge-affiliation selection. **Dual-View Convolution and Jacobi-KAN Classification** propagates information over both hypergraph views and applies a Jacobi polynomial-based KAN-like classifier. An auxiliary classifier attached to the intermediate spectral representation provides additional supervision during training.

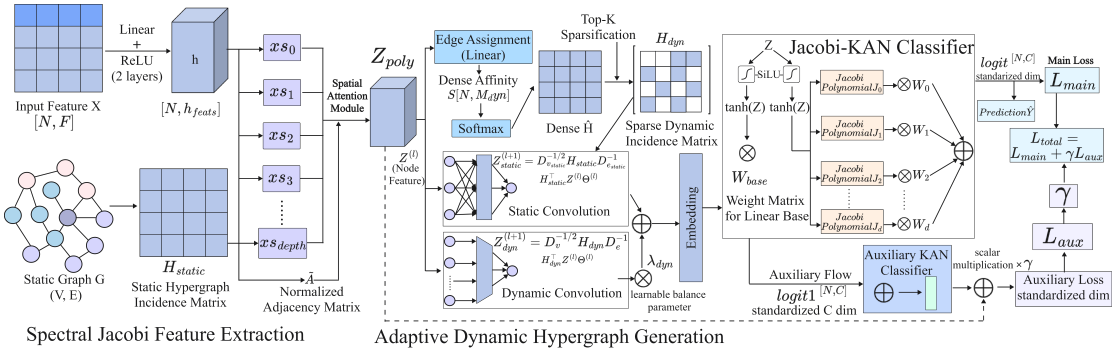


Fig. 1 Overall architecture of JK-DHGNN. The framework contains three components: (1) **Spectral Jacobi Feature Extraction**, which learns the intermediate representation $\mathbf{Z}_{poly}$ from input features and graph topology; (2) **Adaptive Dynamic Hypergraph Generation**, which constructs a sparse dynamic hypergraph incidence matrix $\mathbf{H}_{dyn}$ through Top-K hyperedge affiliation selection; and (3) **Dual-View Convolution and Jacobi-KAN Classification**, in which the static and dynamic branches share $\mathbf{\Theta}^{(l)}$ and are fused using the learnable scalar λ_{dyn} before the final prediction.

4.2 Spectral Jacobi Feature Extraction

Conventional message-passing GNNs often exhibit low-pass filtering behavior. Repeated aggregation can consequently attenuate anomaly-relevant high-frequency components and produce over-smoothing. This effect is particularly problematic when fraudulent nodes connect to benign nodes as a form of camouflage. We therefore introduce a spectral Jacobi graph filter to extract representations at multiple polynomial orders.

The proposed filter uses orthogonal Jacobi polynomials to construct a flexible combination of low- and high-frequency components. Let $\tilde{\mathbf{A}}$ denote the degree-normalized sparse adjacency matrix derived from the observed topology. For an undirected spectral interpretation, directed relations are represented as bidirectional edges before normalization. A learnable feature transformation first maps $\mathbf{X}$ to $\mathbf{X}' \in \mathbb{R}^{N \times F'}$. The filter then recursively computes the order-n representation $\mathbf{Z}_n \in \mathbb{R}^{N \times F'}$. The fixed shape parameters a_{spec} and b_{spec}, selected on the validation set for each dataset, determine the Jacobi basis. The base cases are

$$\mathbf{Z}_0 = \mathbf{X}', \tag{4}$$

$$\mathbf{Z}_1 = \theta_0 \left[\frac{a_{\text{spec}} - b_{\text{spec}}}{2} \mathbf{I} + \frac{a_{\text{spec}} + b_{\text{spec}} + 2}{2} \tilde{\mathbf{A}} \right] \mathbf{X}', \tag{5}$$

where θ_0 is a learnable coefficient. To compute representations for higher-order neighborhoods ($n \geq 2$), we follow a parameterized three-term Jacobi recurrence relation:

$$\mathbf{Z}_n = \theta_{n-1} \left(c_{n,1} \tilde{\mathbf{A}} + c_{n,2} \mathbf{I} \right) \mathbf{Z}_{n-1} - \theta_{n-1} \theta_{n-2} c_{n,3} \mathbf{Z}_{n-2}, \tag{6}$$

where $c_{n,1}$, $c_{n,2}$, and $c_{n,3}$ are the Jacobi recurrence coefficients determined by n, a_{spec}, and b_{spec}. For filtering order d, the recurrence uses the d learnable coefficients

$\{\theta_i\}_{i=0}^{d-1}$. These coefficients are bounded using a hyperbolic tangent function to stabilize optimization and constrain the spectral response.

Filtering to order d produces the representations $\{\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_d\}$. Rather than concatenating or uniformly averaging these representations, a spatial attention module fuses the polynomial orders separately for each node. For order k, an order-specific linear projection is first applied:

$$\mathbf{Z}'_k = \mathbf{Z}_k \mathbf{W}_k, \tag{7}$$

where $\mathbf{W}_k \in \mathbb{R}^{F' \times F'}$ is a learnable projection matrix. Meanwhile, an order-specific global context query $\mathbf{q}_k \in \mathbb{R}^{1 \times F'}$ is obtained by mean-pooling over all N nodes:

$$\mathbf{q}_k = \frac{1}{N} \sum_{v=1}^{N} (\mathbf{Z}_k)_{v,:}. \tag{8}$$

The attention score $s_{k,v}$ is computed from the inner product between the global query and the projected representation of node v, followed by a hyperbolic tangent activation:

$$s_{k,v} = \tanh\left(\mathbf{q}_k \cdot (\mathbf{Z}'_{k,v})^\top\right). \tag{9}$$

The scores are normalized across the $d+1$ polynomial orders using softmax:

$$a_{k,v} = \frac{\exp(s_{k,v})}{\sum_{i=0}^{d} \exp(s_{i,v})}. \tag{10}$$

The representation of node v is then obtained as a weighted combination of the projected order-specific representations:

$$\hat{\mathbf{Z}}_v = \sum_{k=0}^{d} a_{k,v} \mathbf{Z}'_{k,v}. \tag{11}$$

Stacking the node-wise representations $\hat{\mathbf{Z}}_v$ over all nodes yields $\hat{\mathbf{Z}} \in \mathbb{R}^{N \times F'}$. Finally, a linear transformation is applied to obtain the intermediate spectral representation:

$$\mathbf{Z}_{poly} = \hat{\mathbf{Z}} \mathbf{W}_{out}, \tag{12}$$

where $\mathbf{W}_{out}$ is learnable. The node-level attention mechanism allows the contribution of each polynomial order to vary across nodes, providing the representation used for subsequent dynamic hypergraph generation.

4.3 Adaptive Dynamic Hypergraph Generation

The static hypergraph $\mathbf{H}_{static}$ retains the observed pairwise topology but may not explicitly represent latent coordination among multiple entities. We therefore construct a dynamic hypergraph from the intermediate spectral representation $\mathbf{Z}_{poly}$.

First, node representations are projected into a latent hyperedge assignment space:

$$\mathbf{S} = \mathbf{Z}_{poly}\mathbf{W}_{assign}, \tag{13}$$

where $\mathbf{W}_{assign} \in \mathbb{R}^{F' \times M_{dyn}}$ is a learnable weight matrix, and M_{dyn} denotes the number of potential dynamic hyperedges. A row-wise softmax function is then applied to convert the affinity scores into a dense assignment probability matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{N \times M_{dyn}}$:

$$\tilde{\mathbf{H}}_{i,j} = \frac{\exp(\mathbf{S}_{i,j})}{\sum_{m=1}^{M_{dyn}} \exp(\mathbf{S}_{i,m})}. \tag{14}$$

Each row of $\tilde{\mathbf{H}}$ defines a node's distribution over the potential dynamic hyperedges.

A dense assignment matrix can propagate low-confidence affiliations and increase computational cost. We therefore apply **Top-K sparsification**: for each node, only the K largest affiliation scores are retained, and all remaining entries are set to zero:

$$\bar{\mathbf{H}}_{dyn}(i,j) = \begin{cases} \tilde{\mathbf{H}}_{i,j}, & \text{if } \tilde{\mathbf{H}}_{i,j} \in \text{TopK}(\tilde{\mathbf{H}}_{i,:}), \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

The retained scores are subsequently renormalized for each node:

$$\mathbf{H}_{dyn}(i,j) = \frac{\bar{\mathbf{H}}_{dyn}(i,j)}{\sum_{m=1}^{M_{dyn}} \bar{\mathbf{H}}_{dyn}(i,m) + \epsilon}, \tag{16}$$

where ϵ is a small positive constant used for numerical stability. During backpropagation, the Top-K mask is treated as a hard constraint, and gradients pass through the retained scores.

The resulting $\mathbf{H}_{dyn}$ is a sparse, probability-weighted incidence matrix. By removing low-confidence assignments, Top-K sparsification limits their contribution to message propagation. Importantly, K specifies the number of dynamic hyperedge affiliations retained per node; it does not specify the cardinality of an individual hyperedge.

For hypergraph convolution over this weighted structure, we compute the corresponding node and dynamic hyperedge degree matrices as follows:

$$\mathbf{D}_{v,d}(i,i) = \sum_{j=1}^{M_{dyn}} \mathbf{H}_{dyn}(i,j), \tag{17}$$

$$\mathbf{D}_{e,d}(j,j) = \sum_{i=1}^{N} \mathbf{H}_{dyn}(i,j). \tag{18}$$

In the implementation, degree values are lower-bounded by 10^{-5} to prevent division by zero during matrix normalization. Because the retained dynamic affiliations are renormalized row-wise after Top-K sparsification, the dynamic node-degree matrix $\mathbf{D}_{v,d}$ is numerically close to the identity matrix. We retain this term to preserve

a unified notation for the static and dynamic propagation operators, whereas $\mathbf{D}_{e,d}^{-1}$ remains important for controlling differences in dynamic hyperedge mass.

4.4 Dual-View Convolution and Jacobi-KAN Optimization

Given $\mathbf{H}_{static}$ and $\mathbf{H}_{dyn}$, the model performs message passing over both hypergraph views. The static view represents the observed pairwise prior, whereas the dynamic view represents learned higher-order affiliations. Both views share the transformation matrix $\mathbf{\Theta}^{(l)}$ at layer l, reducing the parameter count and encouraging a common feature space.

Given the input representation $\mathbf{Z}^{(l)}$, the static-view propagation is formulated as:

$$\mathbf{Z}_{static}^{(l+1)} = \mathbf{D}_{v,s}^{-1/2}\mathbf{H}_{static}\mathbf{D}_{e,s}^{-1}\mathbf{H}_{static}^{\top}\left(\mathbf{Z}^{(l)}\mathbf{\Theta}^{(l)}\right), \tag{19}$$

where $\mathbf{D}_{v,s}$ and $\mathbf{D}_{e,s}$ denote the node degree matrix and hyperedge degree matrix of the static hypergraph, respectively. Similarly, the dynamic-view propagation is computed as:

$$\mathbf{Z}_{dyn}^{(l+1)} = \mathbf{D}_{v,d}^{-1/2}\mathbf{H}_{dyn}\mathbf{D}_{e,d}^{-1}\mathbf{H}_{dyn}^{\top}\left(\mathbf{Z}^{(l)}\mathbf{\Theta}^{(l)}\right), \tag{20}$$

where $\mathbf{D}_{v,d}$ and $\mathbf{D}_{e,d}$ are the dynamic node- and hyperedge-degree matrices. We employ left node-degree normalization in both the static and dynamic views. The same propagation configuration is used throughout all reported experiments. The two outputs are fused using a learnable scalar λ_{dyn}:

$$\mathbf{Z}_{final}^{(l+1)} = \mathbf{Z}_{static}^{(l+1)} + \lambda_{dyn}\mathbf{Z}_{dyn}^{(l+1)}, \tag{21}$$

where λ_{dyn} is optimized jointly with the remaining model parameters.

After L dual-view convolution layers, prediction is performed using a Jacobi polynomial-based KAN-like classifier. Unlike the B-spline parameterization of the original KAN, this layer represents learnable one-dimensional edge functions using Jacobi polynomial bases and combines them with a SiLU-activated linear branch:

$$\Phi(\mathbf{Z}) = \mathrm{Linear}\left(\mathrm{SiLU}(\mathbf{Z})\right) + \sum_{i=0}^{d_{kan}} J_i\left(\tanh(\mathbf{Z}); a_{\mathrm{kan}}, b_{\mathrm{kan}}\right)\mathbf{W}_i, \tag{22}$$

where $J_i(\cdot; a_{\mathrm{kan}}, b_{\mathrm{kan}})$ denotes the i-th Jacobi basis function, $\mathbf{W}_i$ is its trainable coefficient matrix, and d_{kan} is the maximum degree. The classifier shape parameters are initialized using the dataset-specific spectral values, $a_{\mathrm{kan}}^{(0)} = a_{\mathrm{spec}}$ and $b_{\mathrm{kan}}^{(0)} = b_{\mathrm{spec}}$, and are then optimized during training. By contrast, a_{spec} and b_{spec} remain fixed in the spectral module. The constraints $a_{\mathrm{kan}} > -1$ and $b_{\mathrm{kan}} > -1$ are enforced through

$$a_{\mathrm{kan}} = \mathrm{Softplus}(a'_{\mathrm{kan}}) - 0.99, \quad b_{\mathrm{kan}} = \mathrm{Softplus}(b'_{\mathrm{kan}}) - 0.99, \tag{23}$$

where a'_{kan} and b'_{kan} are learnable variables.

Classifiers of the same functional form are applied to the intermediate representation $\mathbf{Z}_{poly}$ and the final dual-view representation $\mathbf{Z}_{final}^{(L)}$. The former provides auxiliary

supervision, whereas the latter produces the final prediction. The training objective is

$$\mathcal{L}_{total} = \mathcal{L}_{ce}\left(\Phi(\mathbf{Z}_{final}^{(L)})[\mathcal{V}_{tr}], \mathbf{Y}[\mathcal{V}_{tr}]\right) + \gamma\mathcal{L}_{ce}\left(\Phi(\mathbf{Z}_{poly})[\mathcal{V}_{tr}], \mathbf{Y}[\mathcal{V}_{tr}]\right), \tag{24}$$

where $\mathcal{L}_{ce}$ denotes cross-entropy loss, $[\mathcal{V}_{tr}]$ selects the rows corresponding to the training nodes, and γ weights the auxiliary term. Thus, both objectives are computed exclusively on the training mask: the auxiliary objective directly supervises the spectral representation, while the primary objective optimizes the final dual-view embedding.

4.5 Complexity Analysis

We analyze the time and space complexity of JK-DHGNN. Let N be the number of nodes, E_{static} the number of observed static edges, F the hidden dimension, d the Jacobi filter order, L the number of dual-view convolution layers, M_{dyn} the number of candidate dynamic hyperedges, K the number of affiliations retained per node, C the number of classes, and d_{kan} the classifier polynomial degree.

The Spectral Jacobi Feature Extraction module computes multi-order polynomial representations through sparse graph propagation and order-specific feature projection. Computing representations up to order d incurs a time complexity of

$$\mathcal{O}(dE_{static}F + dNF^2). \tag{25}$$

The first term corresponds to sparse matrix multiplications over the graph topology, while the second term comes from order-specific linear projections and attention-based fusion.

The Adaptive Dynamic Hypergraph Generation module first projects node representations into M_{dyn} potential dynamic hyperedges, requiring

$$\mathcal{O}(NM_{dyn}F). \tag{26}$$

Row-wise softmax requires $\mathcal{O}(NM_{dyn})$ operations. Under a heap-based selection implementation, retaining the largest K affiliations for every node requires

$$\mathcal{O}(NM_{dyn}\log K). \tag{27}$$

The exact constant and selection cost depend on the Top-K implementation; when M_{dyn} and K are bounded, this stage scales linearly with N.

For dual-view hypergraph convolution, the static view is based on the sparse incidence structure converted from observed graph edges, while the dynamic view is implemented with dense matrix operations for efficient GPU parallelization. The dominant cost of dynamic-view propagation across L layers is

$$\mathcal{O}(LNM_{dyn}F). \tag{28}$$

The Jacobi-KAN classification layer evaluates Jacobi polynomial bases and applies trainable polynomial coefficients, introducing a complexity of

$$\mathcal{O}(NFCd_{kan}). \tag{29}$$

Combining the above components, the overall time complexity can be summarized as

$$\begin{aligned}\mathcal{O}\big(&dE_{static}F + dNF^2 + NM_{dyn}F \\ &+ NM_{dyn}\log K + LNM_{dyn}F + NFCd_{kan}\big).\end{aligned} \tag{30}$$

For fixed M_{dyn}, K, d, and d_{kan}, the dominant cost is approximately linear in the number of nodes and observed edges, with additional dependence on the hidden dimension.

In terms of space complexity, the static view requires storage proportional to the observed topology, while the dynamic generator stores the intermediate dense assignment matrix before Top-K sparsification. Therefore, the memory complexity is approximately

$$\mathcal{O}(E_{static} + NM_{dyn} + dNF + LNF). \tag{31}$$

The dNF term accounts for retaining the order-specific Jacobi representations for attention-based fusion, and LNF accounts for intermediate layer representations. Although Top-K sparsification yields a sparse final incidence matrix, the current implementation stores the intermediate assignment matrix densely for GPU computation. This $N \times M_{dyn}$ tensor may become the principal memory bottleneck on very large graphs. Mini-batch hypergraph construction, subgraph sampling, or locality-sensitive hashing could reduce this cost.

5 Experiments

This section evaluates JK-DHGNN through five research questions (RQs):

- **RQ1 (Overall Performance):** How does JK-DHGNN compare with representative baselines across fraud detection scenarios?
- **RQ2 (Component and Hyperparameter Analysis):** How do the proposed modules and key hyperparameters affect performance across datasets with different graph characteristics?
- **RQ3 (Structural Stability):** How stable is JK-DHGNN under random structural perturbation?
- **RQ4 (Transductive Time-Aware Case Study):** How does the model perform under a transductive time-aware split on Elliptic?
- **RQ5 (Computational Efficiency):** What are the parameter, training-time, and memory costs of the model?

5.1 Experimental Setup

5.1.1 Datasets

We evaluate JK-DHGNN on three public fraud detection benchmarks: **YelpChi**, **Amazon**, and **Elliptic**. These datasets cover review spam, fraudulent-user, and illicit-transaction detection, respectively. Their statistics are summarized in Table 1.

- **YelpChi (Spam Review Detection) [22]:** YelpChi contains hotel and restaurant reviews. Each node represents a review, and the objective is to distinguish spam from legitimate reviews. The graph includes R-U-R relations between reviews posted by the same user, R-T-R relations between reviews of the same product within the same month, and R-S-R relations between reviews with the same rating. We merge these relations into a homogeneous topology and convert the resulting edges into the static incidence structure.
- **Amazon (Fraudulent User Detection) [23]:** The Amazon graph represents users in the product-review domain and targets users who manipulate ratings. Its U-P-U, U-S-U, and U-V-U relations are merged into a homogeneous topology and converted into the static hypergraph view. Following the dataset protocol, training, validation, and test splits are constructed from the labeled nodes only.
- **Elliptic (Illicit Transaction Detection) [24]:** Elliptic is a Bitcoin transaction graph in which nodes denote transactions and directed edges denote transaction flows. Transactions with unknown labels are removed, and the remaining labeled transactions form an induced subgraph containing 46,564 nodes and 36,624 original directed transaction edges. We retain 93 local transaction attributes after excluding the time-step identifier; the time step is used only to construct the transductive time-aware case-study split. Node indices and edges are remapped after filtering. For graph propagation, each directed edge is represented bidirectionally, resulting in 73,248 directed adjacency entries before normalization, as described in Section 4.2. The primary benchmark uses a stratified random split, while an additional transductive time-aware case study is reported in Section 5.8.

Table 1 Characteristics of the real-world datasets.

Dataset	**YelpChi**	**Amazon**	**Elliptic**
# Nodes	45,954	11,944	46,564
# Edges	3,846,979	4,398,392	36,624[a]
# Features	32	25	93
Anomaly (%)	14.53	9.50	9.76

[a] Elliptic reports original directed transaction edges; bidirectional propagation uses 73,248 directed adjacency entries.

YelpChi and Amazon are dense, multi-relational review graphs, whereas Elliptic is a comparatively sparse transaction-flow graph. Their different densities and semantics allow the model to be evaluated under varied forms of structural noise, class imbalance, and coordinated behavior.

5.1.2 Evaluation Metrics

Because fraudulent nodes constitute a minority class, accuracy can obscure poor fraud detection performance. We therefore report the area under the receiver operating

characteristic curve (**AUC-ROC**), **Macro-F1**, and average precision (**AP**). AUC-ROC evaluates ranking performance across classification thresholds, Macro-F1 weights the two classes equally, and AP summarizes the precision-recall trade-off for the positive fraud class. A fixed decision threshold of 0.5 converts predicted fraud probabilities into class labels for Macro-F1 calculation; this threshold is not optimized on the test set.

5.1.3 Baselines

We compare JK-DHGNN with 16 baselines grouped into four categories:

(1) Conventional GNNs: GCN [8], **GAT** [25], and **GraphSAGE** [9] provide standard neighborhood-aggregation baselines.

(2) Spectral and Structure-learning Methods: GPRGNN [19] and **FAGCN** [26] model adaptive frequency responses, while **IDGL** [27] jointly refines graph structure and node representations.

(3) Fraud-aware and Specialized GNNs: CARE-GNN [10] and **FRAUDRE** [12] target camouflage, heterophily, or class imbalance. We also include **AMNet** [28], **GHRN** [29], **F2GNN** [30], **SparseGAD** [31], **GDN** [32], and **AHFAN** [33] as recent graph anomaly detection methods.

(4) Hypergraph Baselines: HGNN [14] and **HyperGCN** [15] assess the benefit of dynamic Top-K structure learning relative to static hypergraph propagation.

5.1.4 Implementation Details and Hyperparameters

Table 2 summarizes the implementation and hyperparameter settings. JK-DHGNN and the baselines are implemented in Python 3.9 with PyTorch 2.0 and evaluated on a Linux server equipped with an NVIDIA RTX 4090 GPU (24 GB).

Each dataset is partitioned into training, validation, and test sets in an approximate 40%/20%/40% ratio using stratified random sampling. For Amazon, this split is applied only to the labeled subset. Elliptic uses the same random-split protocol to maintain consistency across the benchmark comparison.

We optimize JK-DHGNN using Adam for at most 1000 epochs and apply early stopping with a patience of 150 epochs according to validation AP. The reported objective uses unweighted cross-entropy for both the primary and auxiliary classifiers. The hidden dimension and number of potential dynamic hyperedges are both 64. The dynamic-view coefficient λ_{dyn} is learnable, while the number of hypergraph layers and Jacobi-KAN polynomial degree are fixed at $L = 1$ and $d_{kan} = 3$, respectively. Dataset-specific hyperparameters are selected exclusively according to validation AP. Each configuration is evaluated with model seeds 2023–2027 on the split generated using seed 2023, and results are reported as mean ± sample standard deviation over five runs. After hyperparameter selection, the test set is used only for final evaluation.

All baseline methods use the same train/validation/test split files and the same five model seeds as JK-DHGNN. Each baseline is independently trained for five runs, and its hyperparameters are selected according to validation performance using the official implementation or settings recommended in the original study. The test labels are not used for model selection.

Table 2 Implementation details and dataset-specific hyperparameters.

Category	Hyperparameter	YelpChi	Amazon	Elliptic
General Setup	Split Seed	2023		
	Model Seeds	2023, 2024, 2025, 2026, 2027		
	Number of Runs	5		
	Data Split (Train/Val/Test)	40%/20%/40%		
	Hardware Environment	RTX 4090 (24GB)		
	Software Framework	PyTorch 2.0		
Optimization	Optimizer	Adam		
	Maximum Epochs	1000		
	Early Stopping Patience	150		
	Validation Monitor	AP		
	Initial Learning Rate (lr)	0.01		
	Weight Decay (wd)	5×10^{-4}		
JK-DHGNN	Hidden Embedding Dimension	64	64	64
	Number of Dynamic Hyperedges	64	64	64
	Dropout Rate	0.0	0.0	0.0
	Jacobi Polynomial Order	10	10	10
	Spectral Jacobi Shape a_{spec}	-0.50	0.50	0.50
	Spectral Jacobi Shape b_{spec}	0.00	0.00	0.00
	Auxiliary Loss Coefficient γ	5.0	5.0	5.0
	Top-K Affiliations per Node	10	8	20
	Hypergraph Layers L	1	1	1
	Jacobi-KAN Degree d_{kan}	3	3	3

5.2 Overall Performance Comparison (RQ1)

Table 3 reports node-classification performance on the three datasets. The dataset-specific JK-DHGNN configurations in Table 2 are selected according to validation-set average precision, whereas the test sets are reserved for final evaluation. The highest mean result for each metric is shown in **bold**. Three observations follow from these results.

First, the conventional GNN baselines perform relatively poorly on the review graphs, particularly in AP. GCN, for example, achieves AP values of 25.05% on YelpChi and 39.01% on Amazon. These results indicate that unmodified neighborhood aggregation does not adequately address heterophily and minority-class detection in these datasets.

Second, fraud-aware and structure-learning methods, including IDGL, SparseGAD, GDN, and AHFAN, generally improve upon the conventional baselines. On the two review-based graphs, their pairwise representations or static refinement mechanisms do not explicitly model latent group-level affiliations, which may partly explain the remaining AP gap.

Table 3 Overall performance comparison (%). Results are reported as mean ± sample standard deviation over five runs, and the highest mean result for each metric is highlighted in **bold**.

Methods	YelpChi			Amazon			Elliptic		
	AUC	AP	Macro-F1	AUC	AP	Macro-F1	AUC	AP	Macro-F1
GCN	60.97±0.29	25.05±0.20	48.10±1.37	83.61±0.02	39.01±0.03	47.51±0.00	92.00±0.18	75.20±0.83	76.14±3.26
GAT	73.95±3.34	40.34±3.03	56.10±4.07	94.29±0.98	83.63±1.91	90.39±0.65	90.04±0.27	53.56±1.86	55.89±1.67
GraphSAGE	80.51±0.86	46.51±1.74	59.97±1.30	92.45±0.51	82.91±0.60	89.40±1.38	91.69±0.15	71.31±1.83	64.07±6.69
HGNN	83.05±1.20	51.67±2.03	63.95±2.81	97.50±0.21	88.76±0.54	92.00±0.30	94.25±0.27	79.35±0.65	87.02±0.55
HyperGCN	62.42±0.32	25.67±0.30	49.04±1.04	88.97±0.74	59.40±1.91	66.64±2.52	92.18±0.11	75.31±0.54	79.35±0.40
GPRGNN	83.32±0.55	51.81±1.24	60.55±1.02	95.93±0.17	88.06±0.16	91.29±0.18	94.05±0.23	82.71±0.58	87.08±0.73
FAGCN	85.54±0.64	58.06±1.72	71.32±1.30	95.66±0.35	87.06±0.11	91.43±0.11	94.15±1.40	82.25±3.33	86.21±2.02
CARE-GNN	76.71±0.04	36.71±0.14	59.99±2.49	94.79±0.08	85.53±0.20	88.27±0.58	94.38±0.13	70.42±1.66	76.28±1.12
FRAUDRE	76.22±0.99	37.85±2.06	60.67±1.77	94.25±0.55	85.59±0.50	87.76±1.24	91.30±1.02	68.03±5.61	73.78±2.56
IDGL	85.16±0.43	56.94±1.07	71.85±0.90	97.66±0.15	89.56±0.73	91.85±0.31	94.82±0.95	74.16±10.63	89.88±1.23
AMNet	84.24±0.38	31.91±0.98	69.37±0.73	97.77±0.13	77.60±0.71	**92.80±0.32**	96.56±0.16	74.84±0.87	91.76±0.29
GHRN	85.04±0.40	34.84±0.90	72.41±0.56	97.67±0.25	74.36±2.27	91.81±0.71	97.90±0.08	77.90±0.84	**92.91±0.23**
SparseGAD	86.27±0.83	59.43±1.05	73.17±0.97	97.41±0.47	90.29±1.23	92.08±0.24	97.29±0.16	89.98±0.51	91.51±0.36
F2GNN	84.62±0.34	33.49±0.50	71.50±0.37	98.02±0.11	77.09±2.05	**92.80±0.67**	96.30±0.33	73.30±1.95	91.36±0.66
GDN	87.15±0.19	61.25±0.51	70.44±0.98	97.26±0.28	88.54±0.72	90.28±0.70	**98.04±0.09**	**92.03±0.31**	91.13±0.41
AHFAN	86.58±1.10	61.51±2.27	73.43±1.40	97.52±0.45	91.81±0.71	92.71±0.43	97.18±0.10	89.60±0.46	91.45±0.21
JK-DHGNN (Ours)	**88.87±0.27**	**66.70±0.87**	**75.15±1.00**	**98.15±0.31**	**92.30±0.43**	92.35±0.19	94.90±0.24	84.92±1.55	89.43±0.58

Third, JK-DHGNN demonstrates particularly strong performance on the two review-based datasets. On YelpChi, it achieves the highest mean AUC and AP values of 88.87% and 66.70%, respectively, while also obtaining a Macro-F1 of 75.15%. On Amazon, JK-DHGNN achieves the highest mean AUC and AP values of 98.15% and 92.30%, whereas its Macro-F1 of 92.35% remains competitive with the strongest baselines. These results indicate that the proposed framework is effective for dense review graphs characterized by multiple relation types and structural camouflage.

On Elliptic, JK-DHGNN obtains 94.90% AUC, 84.92% AP, and 89.43% Macro-F1. Although these results outperform several conventional GNN and hypergraph baselines, they remain below specialized anomaly-detection methods such as GDN, SparseGAD, and GHRN on one or more metrics. This result suggests that the complete combination of spectral filtering, dynamic hypergraph learning, and KAN-like classification is less suitable for the sparse transaction-flow structure of Elliptic than for the two dense review graphs.

Overall, the results demonstrate that JK-DHGNN is particularly competitive on review-based fraud detection, while its performance varies according to graph density, relation semantics, and structural characteristics.

5.3 Ablation Study (RQ2)

To assess the contribution of each component, we evaluate four ablated variants on all datasets: "Without Jacobi-KAN" replaces the Jacobi-KAN classifier with a parameter-matched multilayer perceptron; "Without Jacobi Filter" removes spectral filtering; "Without Dynamic Hypergraph" retains only the static view; and "Without Auxiliary Supervision" retains the complete architecture but removes the intermediate auxiliary objective by setting $\gamma = 0$. Table 4 reports the results.

Table 4 Ablation study (%). Results are reported as mean ± sample standard deviation over five runs.

Architecture	YelpChi			Amazon			Elliptic		
	AUC	AP	Macro-F1	AUC	AP	Macro-F1	AUC	AP	Macro-F1
JK-DHGNN	88.87±0.27	66.70±0.87	75.15±1.00	**98.15±0.31**	**92.30±0.43**	92.35±0.19	94.90±0.24	84.92±1.55	89.43±0.58
Without Jacobi-KAN	87.99±0.71	65.34±1.07	74.89±0.85	98.05±0.21	91.37±0.91	92.28±0.25	95.76±0.53	86.64±1.48	90.22±1.25
Without Jacobi Filter	86.95±0.36	62.41±0.77	74.56±0.58	97.37±0.05	90.94±0.18	**92.39±0.30**	**96.56±0.28**	**88.14±1.15**	**91.16±0.55**
Without Dynamic Hypergraph	**89.14±0.45**	**66.98±1.11**	**75.55±0.56**	97.92±0.43	91.77±0.98	92.02±0.62	88.67±1.00	53.26±1.80	72.43±1.38
Without Auxiliary Supervision	85.83±1.00	60.33±2.69	70.19±1.22	96.38±0.66	89.52±0.71	92.02±0.91	93.63±1.19	80.79±3.81	86.52±1.97

Table 4 reveals that the contributions of the proposed components are dataset dependent rather than uniformly positive across all graph types.

On YelpChi, removing the Jacobi filter reduces AP from 66.70% to 62.41%, while removing auxiliary supervision reduces AP to 60.33% and Macro-F1 to 70.19%. Replacing the Jacobi-KAN classifier with a parameter-matched MLP also causes a moderate AP reduction to 65.34%. In contrast, removing the dynamic hypergraph produces a small mean improvement, increasing AP from 66.70% to 66.98%. This suggests that the dense observed relations in YelpChi already provide strong structural information and that the additional learned affiliations may introduce limited redundancy.

On Amazon, the complete model achieves the highest mean AUC and AP. Removing the Jacobi filter decreases AP by 1.36 percentage points, replacing Jacobi-KAN decreases AP by 0.93 points, and removing the dynamic hypergraph decreases AP by 0.53 points. Auxiliary supervision provides the largest contribution, as its removal reduces AP from 92.30% to 89.52%.

Elliptic exhibits a substantially different behavior. Removing the dynamic hypergraph causes AP to decrease sharply from 84.92% to 53.26%, demonstrating that learned high-order affiliations are critical for this sparse transaction graph. Auxiliary supervision also provides a clear benefit. However, removing the Jacobi filter or Jacobi-KAN classifier improves the mean performance, with the Without Jacobi Filter variant reaching 88.14% AP. These results suggest that simultaneously applying multiple polynomial transformations may introduce excessive transformation or optimization interference on sparse transaction-flow graphs.

Overall, the ablation results show that dynamic high-order structure, spectral filtering, nonlinear classification, and auxiliary supervision contribute differently depending on graph density and relation semantics.

5.4 Top-K Sensitivity Analysis (RQ2)

We vary the number of dynamic hyperedge affiliations retained per node over $K \in \{5, 8, 10, 15, 20\}$. Figure 2 reports the best validation AP obtained during training. Points denote the mean over five model seeds, error bars denote the sample standard deviation, and stars indicate the dataset-specific values used in the main experiments.

The preferred sparsity level is dataset dependent. YelpChi reaches its highest mean validation AP of 66.12% at $K = 10$, while Amazon reaches 89.83% at $K = 8$. Their comparatively narrow performance ranges indicate moderate sensitivity to K. Elliptic obtains its highest validation AP of 87.14% at $K = 20$, suggesting that its sparse transaction topology benefits from retaining more learned affiliations. These results

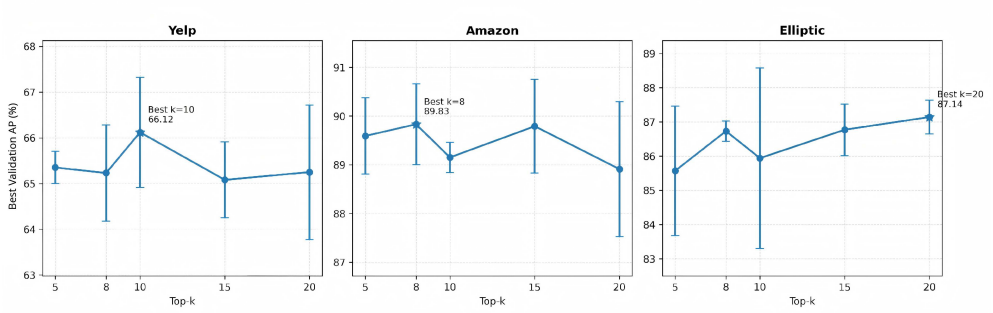


Fig. 2 Sensitivity of validation AP to the Top-K sparsification threshold on YelpChi, Amazon, and Elliptic. Stars identify the selected values: $K = 10$ for YelpChi, $K = 8$ for Amazon, and $K = 20$ for Elliptic.

support dataset-specific selection of K while not isolating Top-K sparsification from the remaining dynamic-hypergraph generator.

5.5 Representation Disentanglement (RQ2)

We use t-SNE to visualize the learned node representations immediately before the final classifier. Figure 3 shows the two-dimensional projections for YelpChi, Amazon, and Elliptic; blue points denote benign nodes and red points denote fraudulent or illicit nodes.

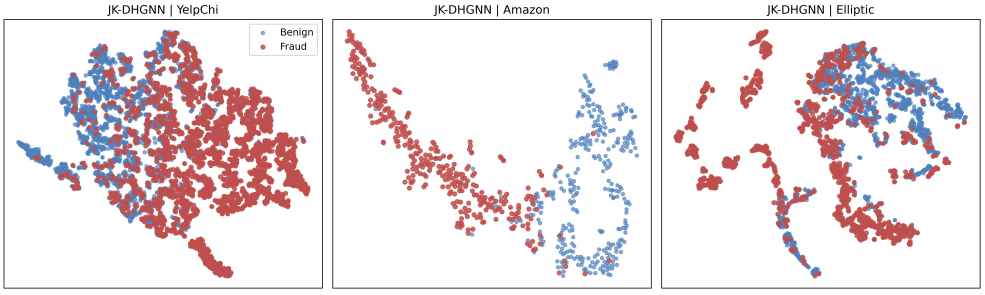


Fig. 3 t-SNE visualization of the learned node embeddings of JK-DHGNN on (a) YelpChi, (b) Amazon, and (c) Elliptic. Blue points denote normal nodes and red points denote abnormal or fraudulent nodes.

The projections show greater visual separation on Amazon and Elliptic. In Figure 3 (b), many fraudulent nodes occupy a concentrated elongated region apart from most benign nodes. In Figure 3 (c), illicit transactions are concentrated along a peripheral trajectory. These qualitative patterns are consistent with the AP and Macro-F1 results in Table 3, although a two-dimensional projection cannot establish class separability in the original representation space.

The YelpChi projection in Figure 3 (a) exhibits greater overlap: fraudulent nodes are dispersed across several local regions rather than forming a single compact cluster. This pattern is compatible with heterogeneous spam behavior, but it should not be interpreted as direct evidence of distinct fraud groups. Overall, the t-SNE plots provide descriptive support for the quantitative results rather than proof of class separability.

5.6 Stability under Random Structural Perturbation (RQ3)

We perturb the observed topology at five edge-injection ratios: 0%, 5%, 10%, 20%, and 30%. For each ratio, non-existing node pairs are sampled without self-loops or duplicate edges and added before training. JK-DHGNN is retrained from scratch on each perturbed topology using model seed 2023 and a fixed perturbation realization. Figure 4 reports the resulting AUC, AP, and Macro-F1 values. These curves represent a descriptive single-seed stability analysis rather than the five-run averages in Table 3; this protocol explains the small difference between the 0% values and the main results.

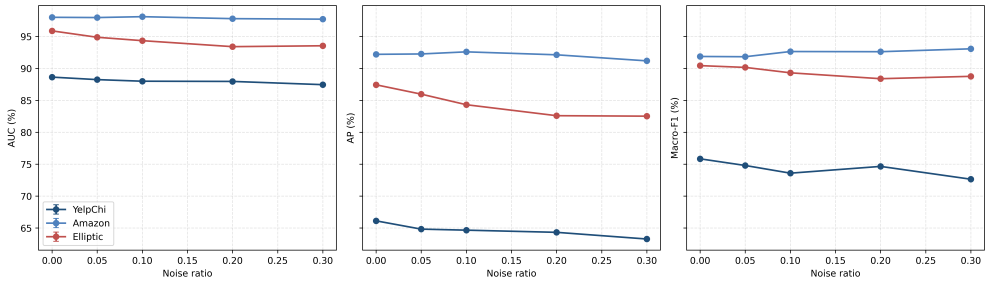


Fig. 4 Performance stability of JK-DHGNN under different ratios of random edge perturbation. Noise edges are injected into the observed topology at ratios from 0% to 30%. Results are obtained with a single model seed and a fixed perturbation realization.

At a 30% random edge-injection ratio, AP decreases from 66.88% to 63.36% on YelpChi, from 92.23% to 91.30% on Amazon, and from 85.29% to 82.46% on Elliptic. The corresponding relative AP reductions are 5.26%, 1.01%, and 3.32%, respectively. These results indicate gradual degradation rather than abrupt failure under random structural perturbation. Because the experiment does not include adversarially optimized attacks or comparative robustness baselines, it should be interpreted as descriptive stability rather than evidence of superior adversarial robustness.

5.7 Sensitivity to Spectral Filtering Order (RQ2)

We further examine the spectral filtering order on the powers-of-two diagnostic grid $d \in \{2, 4, 8, 16, 32\}$ and plot validation AP in Figure 5. The main configuration $d = 10$, reported in Table 2, was selected separately using validation performance and is therefore not part of this diagnostic scaling grid. Accordingly, this experiment characterizes sensitivity across progressively higher polynomial orders rather than repeating the main model-selection sweep. Each point denotes the mean validation AP

over five model seeds, the error bars denote the sample standard deviation, and all other hyperparameters are held fixed. The curves show modest dataset-dependent variation rather than a monotonic collapse, with Amazon and Elliptic remaining comparatively stable and YelpChi exhibiting a somewhat larger but still bounded response range.

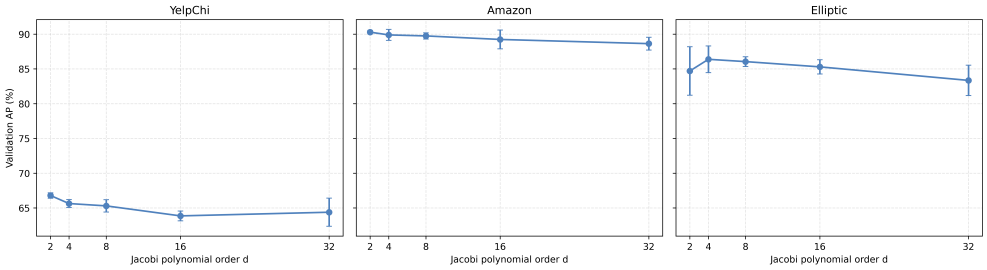


Fig. 5 Validation-AP sensitivity of JK-DHGNN over a powers-of-two diagnostic grid of Jacobi polynomial orders. Points show five-seed means and error bars show sample standard deviations; the separately selected main setting $d = 10$ is not part of this diagnostic grid.

This behavior is consistent with using the Jacobi basis to combine multiple polynomial orders rather than enforcing a purely low-pass propagation pattern. As with the perturbation experiment, the figure should be interpreted as descriptive sensitivity rather than as a direct measurement of representation over-smoothing.

5.8 Transductive Time-Aware Case Study on Elliptic (RQ4)

We additionally conduct a transductive time-aware case study in which labels from the first 60% of time steps are used for training, labels from the following 20% are used for validation, and labels from the final 20% are reserved for testing. The complete graph structure and node features remain available, consistent with the transductive formulation in Section 3.2. The case study uses model seed 2023. As shown in Table 5, JK-DHGNN outperforms GCN and HGNN but remains below GPRGNN in AP and Macro-F1. This result indicates that temporal distribution shift remains challenging for the current architecture. Because the experiment uses one fixed temporal split and model initialization, the results are descriptive and are not intended as statistically conclusive comparisons.

5.9 Computational Efficiency Analysis (RQ5)

JK-DHGNN contains 65,748, 65,300, and 69,652 trainable parameters on YelpChi, Amazon, and Elliptic, respectively. Across five runs, the mean total optimization times to early stopping were 142.13±11.13 s, 165.29±42.27 s, and 33.13±4.79 s, respectively, whereas the corresponding peak CUDA memory usage was 8,263.50±5.58 MB, 9,266.50±1.34 MB, and 910.74±5.80 MB. These results show that JK-DHGNN is compact in terms of trainable parameters; nevertheless, full-batch graph processing and the storage of intermediate structural tensors impose substantial memory overhead on the dense review graphs.

Table 5 A transductive time-aware case study on Elliptic. The results are descriptive and are not intended as statistically conclusive comparisons.

Model	AUC	AP	Macro-F1
JK-DHGNN	81.91	43.55	73.12
GCN	78.03	37.11	49.54
HGNN	75.20	18.30	58.62
GPRGNN	**82.30**	**48.96**	**78.23**

6 Discussion

The ablation results demonstrate that the contributions of the proposed components are strongly dataset dependent. On YelpChi and Amazon, Jacobi filtering and auxiliary supervision yield measurable improvements in AP. By contrast, the dynamic hypergraph branch provides only marginal benefits on YelpChi but is crucial on Elliptic, where its removal markedly degrades predictive performance. Conversely, removing the Jacobi spectral filter or replacing the Jacobi-KAN classifier improves the results on Elliptic. This pattern suggests that simultaneously applying polynomial transformations in the spectral encoder and classifier may introduce redundant feature transformations or optimization interference on sparse transaction graphs. Additional controlled experiments are required to verify this explanation and distinguish architectural effects from dataset-specific optimization behavior.

Taken together, these findings indicate that component effectiveness is conditioned by both graph density and relation semantics. The dense, multi-relational review graphs provide abundant observed connections, enabling spectral filtering and auxiliary supervision to improve minority-class ranking. In the comparatively sparse Elliptic transaction graph, learned higher-order affiliations help compensate for limited observed connectivity, whereas the fixed combination of multiple polynomial modules appears less transferable. Consequently, dataset-adaptive gating or module-selection mechanisms may offer a more reliable extension than applying an identical architectural configuration across heterogeneous fraud-detection scenarios.

This study has several limitations. First, the primary benchmarks use stratified random splits and therefore do not fully emulate forward-looking fraud detection under temporal distribution shift. Second, the time-aware evaluation is transductive and relies on a single fixed split and model initialization; its results should therefore be regarded as descriptive rather than as evidence of prospective or statistically robust performance. Third, random edge injection represents generic structural noise and does not reproduce an adversarially optimized camouflage attack. Finally, constructing the dense intermediate assignment matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{N \times M_{dyn}}$ remains a substantial memory bottleneck for full-batch training on large graphs. Future evaluations should incorporate repeated or rolling temporal splits, multiple model seeds, adaptive structural attacks, and sparse or mini-batch dynamic-hypergraph construction.

7 Conclusion

This study introduced JK-DHGNN, a dual-view hypergraph framework that integrates observed pairwise relations, task-adaptive higher-order affiliations, Jacobi spectral filtering, and KAN-like nonlinear classification for graph-based fraud detection.

Across the stratified random-split benchmarks, JK-DHGNN achieves the highest mean AUC and AP on the YelpChi and Amazon review graphs among the evaluated methods. On Elliptic, it remains competitive with conventional GNN and hypergraph baselines but does not surpass several specialized graph-anomaly-detection approaches. These results indicate that the proposed architecture is particularly effective for dense review graphs, while its relative advantage decreases on sparse transaction networks.

The component analysis further confirms that no single architectural configuration is uniformly optimal across datasets. Jacobi filtering and auxiliary supervision are more beneficial on the review-based datasets, whereas dynamic hypergraph learning provides the largest contribution on the sparse Elliptic graph. The performance gains observed after removing the Jacobi filter or Jacobi-KAN classifier on Elliptic emphasize the importance of dataset-adaptive module integration and careful validation of interacting polynomial transformations.

The random-perturbation analysis indicates gradual rather than abrupt degradation under structural noise, whereas the transductive time-aware case study reveals persistent limitations under distribution shift across time. Future work will investigate adaptive module selection, memory-efficient dynamic-hypergraph construction, repeated temporal evaluation, and stronger adversarial protocols to improve the robustness and practical applicability of JK-DHGNN.

Declarations

Author contributions

D.L. conceived the study, developed the methodology and software, conducted the formal analysis and investigation, curated the data, prepared the visualizations, and drafted the manuscript. J.S. contributed to the methodology, investigation, validation, and data curation and revised the manuscript. X.Y. contributed to software development, validation, formal analysis, and visualization and revised the manuscript. Z.Q. conceived and supervised the study, contributed to the methodology, provided resources, administered the project, and revised the manuscript. All authors reviewed and approved the final manuscript.

Funding

This work was supported by the Youth Project of the National Social Science Fund of China, entitled “Research on Generative Artificial Intelligence-Driven Governance of Telecommunications Network Fraud” (Grant No. 24CFX092).

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Data availability

Publicly available datasets were analyzed in this study. The datasets can be accessed from the following repositories: YelpChi: https://data.dgl.ai/dataset/FraudYelp.zip; Amazon: https://data.dgl.ai/dataset/FraudAmazon.zip; and Elliptic: https://www.kaggle.com/datasets/ellipticco/elliptic-data-set.

Code availability

The implementation of JK-DHGNN is publicly available in the following GitHub repository: https://github.com/shenjie0916-del/JK-DHGNN.

Competing interests

The authors declare no competing interests.

References

[1] Cheng, D., Zou, Y., Xiang, S. & Jiang, C. Graph neural networks for financial fraud detection: a review. *Frontiers of Computer Science* **19**, 199609 (2025).

[2] Qiu, T. *et al. Association-focused path aggregation for graph fraud detection. Advances in Neural Information Processing Systems*, Vol. 38, 6118–6141 (Curran Associates, Inc., 2025). URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/09265e2568cf7a6ff47b506acbc2c6eb-Abstract-Conference.html.

[3] Zou, Y. & Cheng, D. *Effective high-order graph representation learning for credit card fraud detection. Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, 7581–7589 (International Joint Conferences on Artificial Intelligence Organization, 2024). URL https://doi.org/10.24963/ijcai.2024/839. AI for Good.

[4] Luo, X. Robust financial fraud detection via causal intervention and multi-view contrastive learning on dynamic hypergraphs. *Mathematics* **13**, 4018 (2025).

[5] Chikwendu, I. A. *et al.* Spectrum-constrained and skip-enhanced graph fraud detection: Addressing heterophily in fraud detection with spectral and spatial modeling. *Symmetry* **17**, 476 (2025).

[6] Pan, J. *et al. A label-free heterophily-guided approach for unsupervised graph fraud detection. Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39, 12443–12451 (AAAI Press, 2025).

[7] Batreddy, S., Pamisetty, G., Kothuri, M., Verma, P. & Babu, C. S. *Adaptive neighborhood sampling and KAN-based edge features for enhanced fraud detection.*

2024 IEEE International Conference on Big Data (BigData), 3375–3381 (IEEE, 2024).

[8] Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017). URL https://openreview.net/forum?id=SJU4ayYgl.

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

[10] Dou, Y. *et al. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. Proceedings of the 29th ACM international conference on information & knowledge management*, 315–324 (Association for Computing Machinery, 2020).

[11] Liu, Y. *et al. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. Proceedings of the web conference 2021*, 3168–3177 (Association for Computing Machinery, 2021).

[12] Zhang, G. *et al. FRAUDRE: Fraud detection dual-resistant to graph inconsistency and imbalance. 2021 IEEE International Conference on Data Mining (ICDM)*, 867–876 (IEEE, 2021).

[13] Zhou, D., Huang, J. & Schölkopf, B. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems* **19** (2006).

[14] Feng, Y., You, H., Zhang, Z., Ji, R. & Gao, Y. *Hypergraph neural networks. Proceedings of the AAAI conference on artificial intelligence*, Vol. 33, 3558–3565 (AAAI Press, 2019).

[15] Yadati, N. *et al.* HyperGCN: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems* **32** (2019).

[16] Jiang, J., Wei, Y., Feng, Y., Cao, J. & Gao, Y. *Dynamic hypergraph neural networks. Proceedings of the International Joint Conference on Artificial Intelligence*, 2635–2641 (International Joint Conferences on Artificial Intelligence Organization, 2019).

[17] Defferrard, M., Bresson, X. & Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* **29** (2016).

[18] He, M., Wei, Z., Huang, Z. & Xu, H. BernNet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing*

Systems **34**, 14239–14251 (2021). URL https://proceedings.neurips.cc/paper/2021/hash/76f1cfd7754a6e4fc3281bcccb3d0902-Abstract.html.

[19] Chien, E., Peng, J., Li, P. & Milenkovic, O. *Adaptive universal generalized pagerank graph neural network. International Conference on Learning Representations* (OpenReview.net, 2021).

[20] Wang, X. & Zhang, M. *How powerful are spectral graph neural networks. Proceedings of the 39th International Conference on Machine Learning*, Vol. 162 of *Proceedings of Machine Learning Research*, 23341–23362 (PMLR, 2022). URL https://proceedings.mlr.press/v162/wang22am.html.

[21] Liu, Z. *et al. KAN: Kolmogorov–arnold networks. The Thirteenth International Conference on Learning Representations* (OpenReview.net, 2025).

[22] Rayana, S. & Akoglu, L. *Collective opinion spam detection: Bridging review networks and metadata. Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 985–994 (Association for Computing Machinery, 2015).

[23] McAuley, J. J. & Leskovec, J. *From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. Proceedings of the 22nd international conference on World Wide Web*, 897–908 (Association for Computing Machinery, 2013).

[24] Weber, M. *et al. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. KDD '19 Workshop on Anomaly Detection in Finance*, 1–7 (Association for Computing Machinery, Anchorage, Alaska, USA, 2019). URL https://arxiv.org/abs/1908.02591. arXiv:1908.02591.

[25] Veličković, P. *et al. Graph attention networks. International Conference on Learning Representations* (OpenReview.net, 2018).

[26] Bo, D., Wang, X., Shi, C. & Shen, H. *Beyond low-frequency information in graph convolutional networks. Proceedings of the AAAI conference on artificial intelligence*, Vol. 35, 3950–3957 (AAAI Press, 2021).

[27] Wu, J. *et al. IDGL: An imbalanced disassortative graph learning framework for fraud detection. International Conference on Service-Oriented Computing*, 616–631 (Springer, 2022).

[28] Chai, Z. *et al. Can abnormality be detected by graph neural networks? Proceedings of the International Joint Conference on Artificial Intelligence*, 1945–1951 (International Joint Conferences on Artificial Intelligence Organization, 2022).

[29] Gao, Y. *et al. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. Proceedings of the ACM web conference 2023*, 1528–1538

(Association for Computing Machinery, 2023).

[30] Hu, G., Liu, Y., He, Q. & Ao, X. *F2GNN: An adaptive filter with feature segmentation for graph-based fraud detection. 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6335–6339 (IEEE, 2024).

[31] Gong, Z. *et al. Beyond homophily: Robust graph anomaly detection via neural sparsification. Proceedings of the International Joint Conference on Artificial Intelligence*, 2104–2113 (International Joint Conferences on Artificial Intelligence Organization, 2023).

[32] Gao, Y. *et al.* Revisiting attack-caused structural distribution shift in graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering* **36**, 4849–4861 (2024).

[33] Wang, X., Dou, H., Dong, D. & Meng, Z. Graph anomaly detection based on hybrid node representation learning. *Neural Networks* **185**, 107169 (2025).