# Fraud Detection Model on Social Financial Graph Using Dual-Gated Graph Neural Network (DG-GNN)

**Muhamad Misykat Ali Al Mahdi*, Rinaldi Munir, Dimitri Mahayana**

School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia

*Corresponding author: 23522027@std.stei.itb.ac.id

**Abstract.** Fraud detection on large-scale dynamic graphs poses challenges related to label imbalance, structural noise, and oversmoothing in node representations. This study introduces the Dual-Gated Graph Neural Network (DG-GNN), a graph-based architecture that integrates gating mechanisms both edge-wise during message aggregation and within the node update phase. Unlike attention-based methods, DG-GNN adopts a lightweight strategy to filter noisy information with lower computational overhead. Experiments on the DGraph dataset demonstrate that DG-GNN achieves a test ROC-AUC of $0.8538 \pm 0.0009$, outperforming the GEARSage baseline ($0.8460 \pm 0.0002$), while reducing parameter count by over 60% and memory usage by nearly half. The model also delivers the highest results compared to recent attention-based approaches such as HHSGT and DEDGAT.

**Keywords:** *fraud detection; gating mechanism; graph neural network; homophily; social financial graph.*

## 1 Introduction

Graph-based representations have become increasingly important in fraud detection, particularly within the financial technology domain. DGraph, a large-scale social financial graph dataset introduced by Huang, *et al.* [1] provides a realistic benchmark for fraud detection tasks. The graph comprises over 3.7 million nodes and 4.3 million edges, capturing relationships between borrower accounts and their registered emergency contacts. However, DGraph presents several modeling challenges: (i) a high proportion of background nodes without labels, (ii) substantial missing values in node features, and (iii) a high degree of homophily due to label imbalance. These characteristics complicate the learning process and can lead to oversmoothing, where the node embeddings become indistinguishable, particularly between fraudulent and normal nodes [2].

Graph Neural Networks (GNNs) offer a compelling approach to capturing spatial and temporal dependencies in such complex graphs [3]. By leveraging local connectivity structures, GNNs can aggregate neighborhood information to generate expressive node representations for classification tasks [3]. However, in

Received _____, Revised _____, Accepted for publication _____

fraud detection, where fraudulent accounts often connect to non-fraudulent ones to avoid detection, standard GNN aggregation may amplify noise from homophilic neighbors, resulting in oversmoothing and reduced classification performance [4].

Several recent studies have attempted to address this issue using GNN-based model with attention mechanisms that weigh the importance of neighboring nodes. GNN-based models such as DEDGAT [5], TGTOD [6], and HHSGT [7] have incorporated attention and relational modeling to distinguish homophilic from heterophilic edges. While effective, these methods often increase architectural complexity and computational overhead.

In contrast, GEARSage [8] as current state-of-the-art model, achieves highest ROC-AUC score with a simpler architecture with using only two trainable weights per convolutional layer. However, it does not implement any neighbor-specific weighting mechanism, relying instead on uniform aggregation and a fixed residual coefficient during node updates. This design limits its adaptability to diverse graph structures and causes model performance to depend heavily on the capacity of the two projection weights, which may result in high memory usage.

To address these limitations, this study investigates a novel architecture named Dual-Gated Graph Neural Network (DG-GNN), which introduces gating mechanisms at both the message aggregation and node update stages. Inspired by existing works on gated graph models [9], [10], DG-GNN replaces attention with edge-wise gating to assign importance weights to neighbor messages and introduces a gated update mechanism to adaptively control how much information is integrated from the aggregated neighbors and the node's previous state. This dual gating aims to reduce noise from irrelevant neighbors, mitigate oversmoothing, and enhance robustness. Additionally, the model is designed to maintain computational efficiency while reducing the number of parameters and memory usage compared to baseline models GEARSage.

## 2 Nomenclatures

The following nomenclatures are used throughout this paper:

| | | |
|---|---|---|
| $b$ | = | bias term (scalar or vector) |
| $g_{(v,u)}^{(k)}$ | = | edge-wise gate controlling message from node u to node v at layer k |
| $h$ | = | hidden representation of a node |
| $h^e$ | = | edge feature embedding |

| | | |
|---|---|---|
| $h_v^{(k)}$ | = | hidden representation of node *v* at GNN layer *k* |
| $h_{nv}^{(k)}$ | = | aggregated message from all neighbors of node *v* at layer *k* |
| $m_{(v,u)}^{(k)}$ | = | message from node *u* to node *v* at layer *k* |
| $\widetilde{m}_{(v,u)}^{(k)}$ | = | gated message from node *u* to node *v* at layer *k* |
| $\mathcal{N}(v) \in V$ | = | direct neighbors of node *v* |
| $\mathcal{N}_k(v) \in V$ | = | *k*-hop neighbors of node *v* |
| $W$ | = | weight matrix |
| $x$ | = | node feature vector |
| $x^e$ | = | edge feature vector |
| $x'$ | = | node feature vector after feature engineering |
| $x_{(v,u)}^e$ | = | edge feature vector connecting node *v* and node *u* |
| $\alpha \in [0, 1]$ | = | gate coefficient for node update mechanism |
| $\odot$ | = | element-wise multiplication operator |
| $\|\|$ | = | vector concatenation operator |

## 3 Related Works

**GNN for Fraud Detection.** A range of GNN-based architectures have been developed to address common challenges in fraud detection such as camouflage, oversmoothing, and label imbalance. Most of these methods employ attention mechanisms, popularized by the Graph Attention Network (GAT) [11], to selectively aggregate information from relevant neighbors. For instance, GAS [12] integrates Recurrent Neural Networks (RNN) with GAT to detect account takeover frauds by modeling temporal dependencies. STAGN [13] applies both spatial and temporal attention for credit card fraud detection, enabling the model to focus on time-sensitive transaction behaviors. MAFI [14] addresses fraud detection on heterogeneous graphs by combining attention with intra-relation aggregation, allowing it to distinguish between different types of relationships. SemiGNN [15] adopts a hierarchical attention mechanism to detect financial fraud in multiview graphs, effectively capturing both local and global patterns. Despite their effectiveness, attention-based approaches tend to introduce additional model complexity, which may increase computational costs and amplify noise, especially in large-scale or noisy graph settings.

**Existing Models on Dataset DGraph.** Most of the top-performing models on the DGraph dataset also employ attention mechanisms to improve performance. For instance, HHSGT [7] integrates sparse graph transformers with relation scoring to capture structural fraud patterns. DEDGAT [5] constructs dual node embeddings (incoming and outgoing) at each convolutional layer and applies attention to refine message construction from neighbors. CAFD [16] utilizes

attention to fuse temporal frequency encoding with out-degree encoding. Despite these advanced mechanisms, these models have yet to surpass the performance of GEARSage [8] as the current state-of-the-art model, which adopts a significantly simpler architecture.

GEARSage uses only two trainable weights in each convolutional layer: one for projecting the target node's features and the other for projecting the features of neighboring nodes along with their edge attributes. Unlike attention-based GNN models, GEARSage does not implement any neighbor-specific weighting mechanism. As a result, the model's performance heavily depends on how effectively information is captured by these two projection parameters, which can lead to high training memory consumption. This indicates that there is still room to improve the architecture of GEARSage to reduce training memory consumption while enhancing its overall performance.

**Gated GNN.** Initially proposed in LSTM [17], gating was later extended to convolutional networks through Gated Linear Units (GLU) and its variants [18], which combine transformed inputs with learnable gates using element-wise multiplication and, optionally, residual connections. These principles have been applied in GNNs to enhance message passing. Marcheggiani and Titov [9] introduced edge-wise gating with sigmoid activation to weigh neighbor messages. Bresson and Laurent [19] expanded this by integrating residual connections, improving depth scalability. Jiao *et al.* [10] further refined edge gating in egGNN using layer normalization, GELU, and exponential functions, incorporating edge features and enabling multi-head gating.

These studies demonstrate that gating provides an efficient alternative to attention, especially for controlling noise in large-scale, heterophilic graphs. However, these methods typically rely solely on edge features as gating inputs, without incorporating the features of neighboring nodes as part of the gating function. Moreover, to the best of our knowledge, none of the existing approaches simultaneously apply gating mechanisms both edge-wise during message aggregation and within the node update phase.

**Table 1** Comparison of existing GNN models on social financial fraud dataset DGraph

| Method | Complexity | # Parameters | ROC-AUC |
|---|---|---|---|
| Standard GNN (GCN [20], GraphSAGE [21], GAT [11]) | Low | Low | 0.70-0.78 [5] |
| Attention-based GNN for Fraud (HHSGT [7], CAFD [16], DEDGAT [5]) | High | High | 0.81-0.83 |
| Current SOTA (GearSAGE) [8] | Low | High | 0.846 |
| Dual Gated-based GNN (proposed) | Medium | Low | > 0.846 |

## 4 Proposed Method

This section presents the proposed solution developed as an extension of the GEARSage architecture, introduced by Li *et al.* [8], which serves as the baseline model and demonstrates high performance with relatively low complexity. The key novelty of this research lies in the design of a new architecture called the Dual-Gated Graph Neural Network (DG-GNN), which introduces two levels of gating mechanisms: (1) an edge-wise gating for message filtering from neighboring nodes, and (2) a dynamic gated update mechanism for node feature updates. Furthermore, we propose a modified temporal embedding strategy to improve the representation of timestamp-based edge features. An overview of the proposed architecture is depicted in Figure 1.

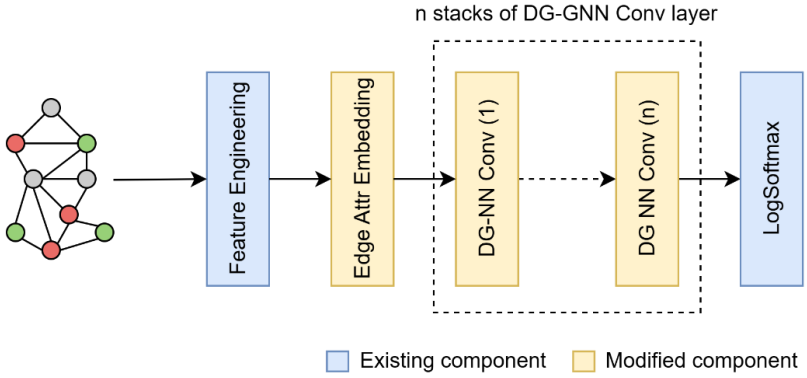


**Figure 1** Proposed enhanced architecture from

### 4.1 Feature Engineering

The input feature vector for each node is constructed by combining multiple engineered components, designed to capture both structural and semantic properties relevant to fraud detection in financial social networks. Importantly, the feature engineering process in this study is directly adopted from the GEARSage model [8] without any modification. The resulting enriched feature vector is denoted as $x' \in \mathbb{R}^{d'}$ and defined by the concatenation:

$$x' = \left[x^{mv} || x^{sim} || x^{deg} || x^{bn} || x^{st} || x^{nt} || x^{la}\right] \quad (1)$$

Each component is described as follows:

1. Missing Value Flag ($x^{mv} \in \mathbb{R}^{34}$). This component encodes both the imputed values and their missingness. The first 17 dimensions contain the original feature values where all missing entries that originally −1 are replaced with 0. The next 17 dimensions are binary flags indicating whether each original feature was missing. The equation is formally written in Eq. (2).

$$x^{mv} = [x^{filled} || x^{flag}], x_{(i)}^{filled} = \begin{cases} x_{(i)}, if\ x_{(i)} \neq -1 \\ 0, otherwise \end{cases}, x_{(i)}^{flag} = \begin{cases} 1, if\ x_{(i)} \neq -1 \\ 0, otherwise \end{cases} \quad (2)$$

2. Neighbor Similarity ( $x^{sim} \in \mathbb{R}^1$). Represents the sum of cosine similarity scores between the target node and each of its neighbors as described in Eq. (3).

$$x^{sim} = \sum_{u \in \mathcal{N}(v)} \cos(x_v, x_u) \quad (3)$$

3. In & Out Degree ($x^{deg} \in \mathbb{R}^2$). Encodes the in-degree and out-degree of node, as described in Eq. (4).

$$x^{deg} = \left[deg^{in}(v), deg^{out}(v)\right] \quad (4)$$

4. Background Neighbor Count ($x^{bn} \in \mathbb{R}^1$). Counts the number of neighbors that belong to the "background" class (labels 2 and 3), as described in Eq. (5).

$$x^{bn} = \left|\{u \in \mathcal{N}(v) \mid y_u \in \{2,3\}\}\right| \quad (5)$$

5. Edge Timestamp Summary ($x^{temp} \in \mathbb{R}^2$). Summarizes the temporal activity of the node by aggregating the timestamps of its incident edges, as described in Eq. (6).

$$x_v^{temp} = \left[\sum_{e \in E_v} x^{e,temp},\ \max\ x^{e,temp}\right] \quad (6)$$

6. Edge Type Frequency ($x^{nt} \in \mathbb{R}^{11}$). A histogram vector counting how many times each of the 11 edge types appears in the neighborhood, as described in Eq. (7).

$$x_{(i)}^{nt} = |\{e \in E_v \mid type(e) = i\}|,\ i = 1, \ldots, 11 \quad (7)$$

7. Edge Type Frequency ($x^{la} \in \mathbb{R}^1$). The original dataset has four labels: 0 (normal), 1 (fraud), and 2–3 (background classes). This feature map the fraud and normal label into 0 and the remaining labels (0, 2, 3) are shifted to (0, 1, 2). Finally, the feature is one-hot encoded to a 3-dimensional vector, as described in Eq. (8).

$$x_{(i)}^{la} = \begin{cases} 1, & if\ adjusted\_label(v) = i \\ 0, & otherwise \end{cases},\ i \in \{0,1,2\} \quad (8)$$

## 4.2 Edge Embedding

Each edge in the graph contains three attributes, as follows:

1. $x^{e,temp} \in \mathbb{R}^1$: timestamp of emergency contact creation
2. $x^{e,type} \in \{0,1\}^{12}$: one-hot encoding of relationship type
3. $x^{e,dir} \in \{0,1\}^2$: direction of the edge (inbound or outbound)

These features then transformed into embeddings as follows:

$$h^{e,temp} = TemporalEncoding(x^{e,temp}) \quad (9)$$

$$h^{e,attr} = EmbeddingType\left(x^{e,dir}\right) + EmbeddingDir(x^{e,type}) \quad (10)$$

$$h^{e} = [h^{e,attr} \,||\, h^{e,temp}] \quad (11)$$

To construct $h^{e,temp}$, we replace the original time encoder using non-trainable temporal encoding from [22] as written in Eq. (12) with *d* as the embedding size. As for the *EmbeddingType* and *EmbeddingDir*, following the baseline model, we adopt a simple lookup table approach as shown in Eq. (13), where *E* denotes the trainable weight matrix, *d* denotes the resulting embedding size and *i* represents the index of the embedded value. As in the previous work, we then define $h^{e,attr}$ by performing element-wise summation on the resulted embeddings of edge type and edge direction. The final edge attribute $h^{e}$ is then constructed by performing concatenation between $h^{e,attr}$ and $h^{e,temp}$.

$$TemporalEncoding(t)_i = cos\left(t \cdot \sqrt{d}^{-(i-1)/\sqrt{d}}\right) \quad (12)$$

$$Embedding(i) = E_i \in \mathbb{R}^d \quad (13)$$

### 4.3 Dual-Gated Graph Neural Network (DG-GNN)

The Dual-Gated Graph Neural Network (DG-GNN) is the core contribution of this research. It extends the message passing framework in GEARSage by introducing two levels of gating mechanisms to explicitly control the flow of information during the learning process. These mechanisms are designed to address the oversmoothing problem, which arises when node representations become indistinguishable due to excessive mixing of neighborhood information that often exacerbated by noisy or irrelevant neighbor features.

The two gating mechanisms are:

1. Edge-Wise Message Gating: filters messages from neighbors before aggregation, based on edge attributes.
2. Residual Gated Update: balances the contribution between the current node state and the aggregated message using a dynamically computed gate.

An overview of the convolution architecture is shown in Figure 2.

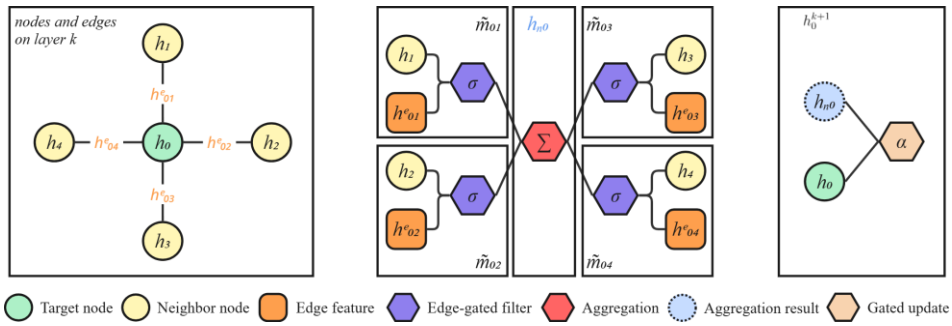


**Figure 2** DG-GNN convolution architecture

### 4.3.1 Message Construction with Edge-Wise Gating

In standard GNNs, all neighbors contribute equally or based on learned attention scores. However, this can be problematic in graphs where some edges carry more semantic importance than others. Inspired by the edge-gated GNN approach [10], our model learns a message-specific gate that depends solely on the edge feature vector $h^e_{(v,u)}$. Each neighbor node $u \in \mathcal{N}(v)$ sends a message to node that is conditioned on both its own representation and the connecting edge, as follows:

$$m^{(k)}_{(v,u)} = \left[h^{(k-1)}_u, h^e_{(v,u)}\right] \cdot W^{(k)}_m + b^{(k)}_m \tag{14}$$

A gating score is computed from the edge embedding:

$$g^{(k)}_{(v,u)} = \sigma\left(h^e_{(v,u)} \cdot W^{(k)}_{ge} + b^{(k)}_{ge}\right) \tag{15}$$

The final gated message becomes:

$$\tilde{m}^{(k)}_{(v,u)} = m^{(k)}_{(v,u)} \odot g^{(k)}_{(v,u)} \tag{16}$$

This gate acts as a soft mask, suppressing or amplifying messages based on edge semantics. The gating mechanism employed in DG-GNN shares similarities with the approach used in egGNN [10], but there are two key differences. First, while egGNN uses the neighbor node feature $h^{(k-1)}_u$ as the gating target, DG-GNN applies the gating function over the message $m^{(k)}_{(v,u)}$, which is the concatenation of the neighbor feature and the edge feature. Second, egGNN combines layer normalization with an exponential activation function to compute the gate value, whereas DG-GNN uses a sigmoid activation function. The gating mechanism in DG-GNN also resembles the approach proposed by Marcheggiani and Titov [9], who similarly use a sigmoid activation for gating. However, like egGNN, their method only considers the neighbor feature $h^{(k-1)}_u$ as the input to the gating function.

### 4.4 Message Aggregation

The filtered messages are aggregated using element-wise summation, as follows:

$$h_{nv}^{(k)} = \sum_{u \in \mathcal{N}(v)} \widetilde{m}_{(v,u)}^{(k)} \tag{17}$$

Summation is chosen for its simplicity and scalability to graphs with highly variable node degrees. Since each incoming message has already been selectively gated, the aggregated result reflects a more robust and noise-reduced neighborhood context.

### 4.5 Node Representation Update with Residual Gated Mechanism

To update the node representation, we propose a residual gating mechanism that dynamically combines the previous node state $h_v^{(k-1)}$ and the aggregated neighborhood message $h_{nv}^{(k)}$.

First, a gate vector $\alpha \in [0, 1]$ is computed as:

$$\alpha = \sigma(W_{gr}^{(k)} \cdot [W_m^{(k)} h_{nv}^{(k)} || \, W_z^{(k)} h_v^{(k-1)}]) \tag{18}$$

The final node embedding is then updated as:

$$h_v^{(k)} = \alpha \cdot h_{nv}^{(k)} + (1 - \alpha) \, h_v^{(k-1)} \tag{19}$$

This gated residual update allows the model to adaptively control how much of the new information should replace or complement the existing state. It provides a finer-grained control than standard residual connections or static blending factors (e.g., GEARSage uses a fixed α). From an information-theoretic standpoint, this mechanism mitigates oversmoothing by preserving node individuality and enabling information decay where necessary—an essential property in heterogeneous or fraud-prone networks. Compared to prior work, DG-GNN, by contrast, integrates both edge information and previous node states into its gating decision, making it more expressive and stable across deeper layers.

## 5 Result and Discussion

This section presents the dataset, experiment design and the results of the experiments conducted on the DG-GNN model. The results are including the impact of hyperparameter tuning, ablation studies to assess the contribution of each component, and a comparison with baseline and related models.

## 5.1 Dataset

The dataset used in this study is DGraph [1], a large-scale financial social graph consisting of 3,700,550 nodes and 4,300,999 edges. The graph represents the relationships between loan applicant accounts and their registered emergency contacts. DGraph is derived from real-world data provided by Finvolution Group. Each node in DGraph is equipped with a 17-dimensional feature vector, describing user demographics and loan history. Due to Finvolution's policy of allowing optional inputs, 49.9% of the feature values are missing. Edges in the graph are temporal, reflecting the most recent emergency contact updates made by users before each loan application. Additional edge attributes such as kinship type are also available, making DGraph a dynamic graph with temporal and semantic relationships. Only 15,509 nodes (0.42%) are labeled as fraudsters, 1,210,092 nodes (32.7%) as normal users, and 2,474,949 nodes (66.88%) as background nodes—accounts with no borrowing history.

## 5.2 Experiment Design

Experiments were conducted on NVIDIA RTX A5000 (24GiB) and Quadro RTX 5000 (16GB). The node classification task in this study follows a semi-supervised learning setting, where all node types (training, validation, and test) are included during training to preserve graph structure and relation, unlike traditional supervised learning. The experiments are repeated 10 times (runs), each consisting of up to 500 epochs. Early stopping is applied if validation ROC-AUC does not improve for 100 epochs. Each epoch consists of:

1. Undersampling: A subgraph $G' = (V', E')$ is sampled to balance fraud and normal labels.
2. Training: Model receives $G'$, outputs prediction, computes training loss via NLLLoss, and ROC-AUC.
3. Validation: Inference on full graph $G$ to obtain prediction, then computes validation loss and ROC-AUC.

The same undersampling strategy as the baseline model is applied, as follows:

$$\mathcal{N}^3(V_{base}) = \bigcup_{v \in V_{base}} \mathcal{N}^3(v) \tag{20}$$

$$V' = V_{base} \cup \mathcal{N}^3(V_{base}) \tag{21}$$

$$E' = \{(u, v) \in E \mid u \in V', v \in V'\} \tag{22}$$

$$G' = (V', E') \subset G = (V, E) \tag{23}$$

Where

$$V_{base} = V_{fraud}^{train} \cup \tilde{V}_{normal}^{train}$$

## 5.3 Hyperparameter Tuning Result

First, we conducted an experiment to determine the optimal set of hyperparameters using a sequential hyperparameter tuning strategy. This strategy involves tuning one hyperparameter at a time while keeping the others fixed, typically using default or previously selected values. For each hyperparameter, several candidate values were evaluated, and the value that yielded the highest ROC-AUC score on the validation set was selected. The selected value for each hyperparameter is indicated with an asterisk (*) in Table 2. Once the best value was determined for one hyperparameter, it was fixed, and the next hyperparameter was tuned in the same manner. This process was repeated iteratively until all hyperparameters had been explored. The results of each tuning step are visualized in Figure 4.

**Table 2** Hyperparameter values

| No | Hyperparameter | Values |
|---|---|---|
| 1 | Hidden Size | 30, 40*, 45, 50, 60, 70, 80 |
| 2 | Weight Decay | 1e-5, 7e-5*, 1e-4, 2e-4, 1e-3 |
| 3 | Activation Function | ReLU, ELU* |
| 4 | Learning Rate | 1e-4, 1e-3*, 1e-2 |
| 5 | Edge Embedding Size (Edge Attr & Edge Timestamp) | 10 & 20*, 20 & 20, 30 & 30 |
| 6 | Dropout | 0.2 & 0.3* |
| 7 | Number of Conv Layers | 3*, 4, 5 |
| 8 | Mekanisme Gating | GLU*, GTU, GTRU |

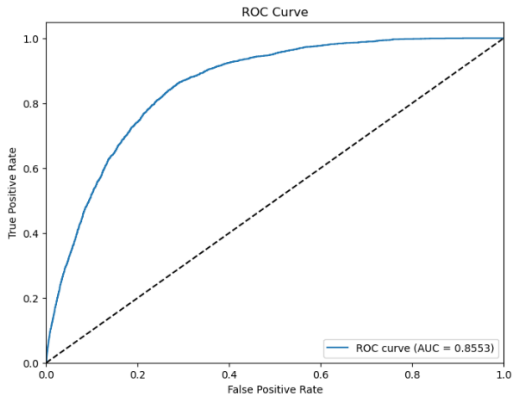


**Figure 3** ROC-AUC curve of the best performing model

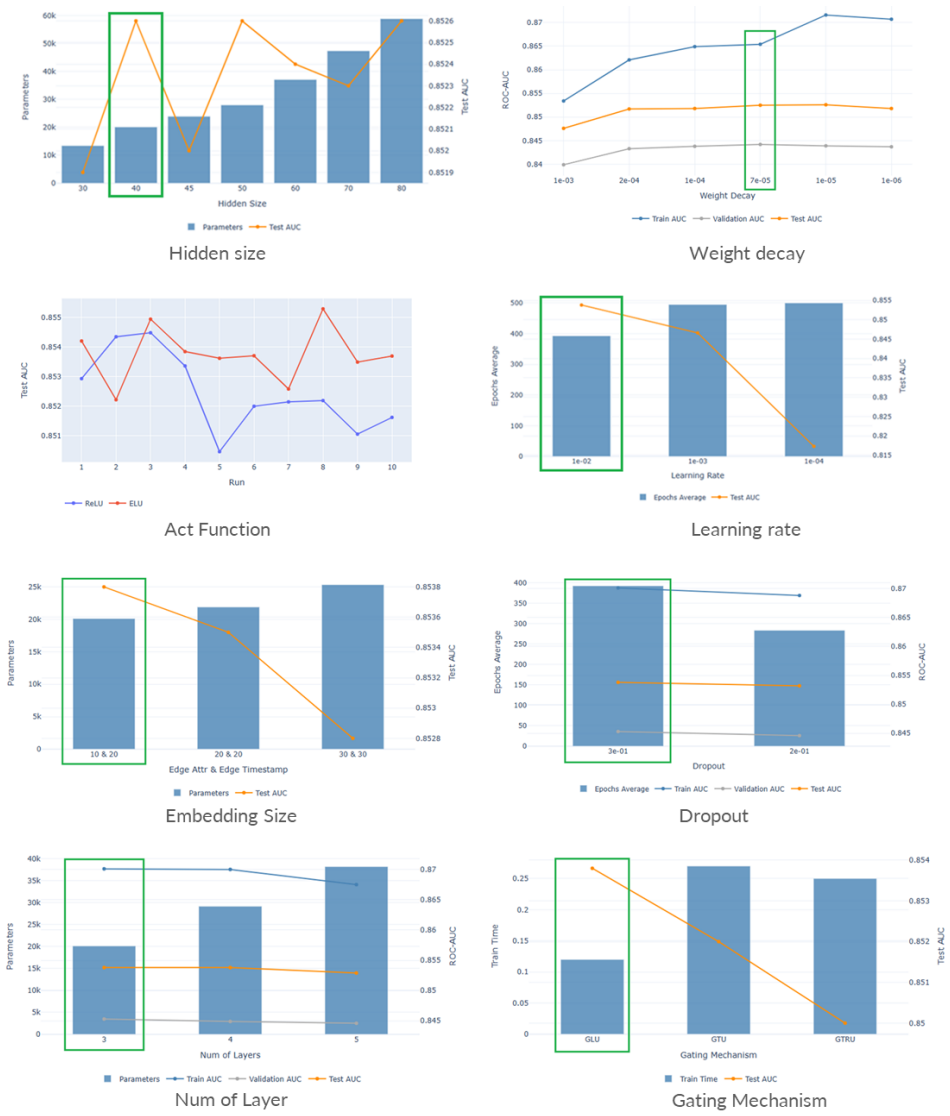


**Figure 4** Hyperparameter tuning result

The selected hyperparameters yielded a ROC-AUC score of 0.8538 ± 0.0009, representing an improvement of 0.0078 over the baseline model, while also reducing the number of parameters by 40% (from 50,544 to 20,126). To evaluate the effect of architectural changes independently, the selected hyperparameters were applied to the baseline model, resulting in a performance drop of up to 0.0057. This confirms that the improvement is attributed not only to hyperparameter tuning but also to the proposed architectural enhancements.

Several key observations were drawn: (1) the model exhibits sensitivity to noise, as evidenced by performance degradation when using larger hidden sizes or ReLU activation; (2) regularization techniques—dropout, weight decay, and early stopping—effectively mitigate overfitting; and (3) DG-GNN benefits from weight decay due to its architectural complexity, whereas the baseline performs better without it. Figure 3 presents the ROC-AUC curve of the best-performing model, which achieved a peak ROC-AUC score of 0.8553.

## 5.4 Ablation Study

An ablation study conducted to evaluate the individual contribution of each component in the proposed DG-GNN architecture and to assess the interactions among these components. All experiments were performed using the optimized hyperparameters obtained through prior tuning.

**Table 3** Ablation study result

| No | Feature Engineering | Time Embedding | Edge-gated | Update-gated | ROC-AUC | △ |
|---|---|---|---|---|---|---|
| 1 | Yes | Temporal Embedding | Yes | Yes | 0,8538 ± 0,0009 | 0 |
| 2 | **No** | **Time Encoding** | **No** | **No** | 0,8299 ± 0,0010 | -0,0239 |
| 3 | Yes | **Time Encoding** | **No** | **No** | 0,8400 ± 0,0007 | -0,0138 |
| 4 | **No** | Temporal Embedding | Yes | Yes | 0,8440 ± 0,0010 | -0,0098 |
| 5 | Yes | **Time Encoding** | Yes | Yes | 0,8509 ± 0,0006 | -0,0029 |
| 6 | Yes | Temporal Embedding | **No** | Yes | 0,8493 ± 0,0006 | -0,0044 |
| 7 | Yes | Temporal Embedding | **2 heads** | Yes | 0,8529 ± 0,0007 | -0,0009 |
| 8 | Yes | Temporal Embedding | **3 heads** | Yes | 0,8528 ± 0,0008 | -0,0010 |
| 9 | Yes | Temporal Embedding | Yes | **No** | 0,8499 ± 0,0004 | -0,0039 |

Table 3 presents all the experimental scenarios. The results show that removing all components reduced the ROC-AUC by 0.0239, while removing only the newly proposed components resulted in a decrease of 0.0138. These findings indicate that both the original components adopted from the baseline model and the newly proposed components contribute significantly to the model's performance.

The feature engineering, adopted directly from the GEARSage baseline, still showed a significant impact with a delta of 0.0098. Temporal Embedding

consistently outperformed the baseline Time Encoding, particularly when combined with the Edge-Gated Convolution, indicating a strong positive interaction. The Edge-Gated Convolution itself contributed a delta of 0.0044, while adding multi-head gating slightly reduced performance, likely due to increased complexity and noise. Lastly, the Gated Update Mechanism provided an improvement of 0.0039 over a fixed-weight residual update. Overall, the results confirm that all components are complementary and non-redundant.

## 5.5 Performance Comparation

To evaluate the performance of DG-GNN, we compare it with the GEARSage baseline model and several existing models reported in related studies. DG-GNN achieves a test ROC-AUC score of 0.8538 ± 0.0009, which is an improvement of +0.0078 over the GEARSage baseline (0.8460 ± 0.0002), while reducing the number of model parameters by more than 60% (from 50,544 to 20,126) and lowering memory usage from 18.5 GB to 9.5 GB.

When compared to other models, such as HHSGT (0.8340), CAFD (0.8150 ± 0.0009), and DEDGAT (0.8137 ± 0.0006), DG-GNN shows higher ROC-AUC scores. It also achieves better performance than recent temporal graph models including TE-GAT (0.791), TGTOD (0.7830 ± 0.0003), and TGN (0.7747), despite using fewer parameters and moderate memory resources. These findings suggest that the combination of dual gating mechanisms and enriched node features contributes to performance improvements on the DGraph dataset.

**Table 4** Model performance comparison

| No | Model | ROC-AUC | Parameters | Training Memory |
|---|---|---|---|---|
| 1 | DG-GNN | 0.8538 ± 0.0009 | 20,126 | 9.5 GB |
| 2 | GEARSage [8] | 0.8460 ± 0.0002 | 50,544 | 18.5 GB |
| 3 | HHSGT [7] | 0.8340 | Unknown | Unknown |
| 4 | CAFD [16] | 0.8150 ± 0.0009 | Unknown | Unknown |
| 5 | DEDGAT [5] | 0.8137 ± 0.0006 | 88,200 | Unknown |
| 6 | TE GAT [23] | 0.791 | Unknown | Unknown |
| 7 | TGTOD [6] | 0.7830 ± 0.0003 | 6,865 | 16 GB |
| 8 | TGN [24] | 0.7747 | Unknown | Unknown |
| 9 | GODM [25] | 0.7580 | Unknown | < 1 GB |

# 6 Conclusions

This study introduces DG-GNN, a Dual-Gated Graph Neural Network designed to tackle challenges in large-scale, dynamic financial graphs with severe label imbalance. The architecture incorporates two gating mechanisms: edge-gated convolution for filtering messages using both node and edge features, and a gated

update mechanism for adaptively integrating neighbor information with prior node states. Ablation studies confirm that both components contribute meaningfully to performance gains.

Experiments on the DGraph dataset show that DG-GNN outperforms the GEARSage baseline and several attention-based models, achieving a test ROC-AUC of $0.8538 \pm 0.0009$—an increase of 0.0078 over the baseline. Remarkably, this improvement is achieved with only 20,126 parameters (40% of baseline's parameter number) and a training memory footprint of 9.5 GB, which is almost half of the memory required by the baseline model. Future work may explore strategies to improve training stability and evaluate the model's adaptability on other large-scale fraud detection datasets with varying degrees of label imbalance and structural heterogeneity.

## References

[1] X. Huang *et al.*, "DGraph: A Large-Scale Financial Dataset for Graph Anomaly Detection," *36th Conf. Neural Inf. Process. Syst.*, vol. 35, 2022.

[2] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang, "Addressing Heterophily in Graph Anomaly Detection: A Perspective of Graph Spectrum," in *Proceedings of the ACM Web Conference 2023*, Austin TX USA: ACM, Apr. 2023, pp. 1528–1538. doi: 10.1145/3543507.3583268.

[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "Survey 2021 A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

[4] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Virtual Event Ireland: ACM, Oct. 2020, pp. 315–324. doi: 10.1145/3340531.3411903.

[5] J. Wu *et al.*, "DEDGAT: Dual Embedding of Directed Graph Attention Networks for Detecting Financial Risk," Mar. 06, 2023, *arXiv*: arXiv:2303.03933. doi: 10.48550/arXiv.2303.03933.

[6] K. Liu, J. Ding, M. Torkamani, and P. S. Yu, "TGTOD: A Global Temporal Graph Transformer for Outlier Detection at Scale," Dec. 01, 2024, *arXiv*: arXiv:2412.00984. doi: 10.48550/arXiv.2412.00984.

[7] X. Wang, L. Xiangfeng, X. Wang, and H. Yu, "Homophilic and Heterophilic-Aware Sparse Graph Transformer for Financial Fraud Detection," in *2024 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2024, pp. 1–8. doi: 10.1109/IJCNN60899.2024.10650212.

[8] J. Li, Z. Yu, W. Sun, X. Jin, Q. Wang, and L. Chen, "GEARSage," Competition Technical Report, 2022. [Online]. Available: https://github.com/storyandwine/GEARSage-DGraphFin/tree/main

[9] D. Marcheggiani and I. Titov, "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling," Jul. 30, 2017, *arXiv*: arXiv:1703.04826. doi: 10.48550/arXiv.1703.04826.

[10] Q. Jiao, Z. Qiu, Y. Wang, C. Chen, Z. Yang, and X. Cui, "Edge-Gated Graph Neural Network for Predicting Protein-Ligand Binding Affinities," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec. 2021, pp. 334–339. doi: 10.1109/BIBM52615.2021.9669846.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," Feb. 04, 2018, *arXiv*: arXiv:1710.10903. doi: 10.48550/arXiv.1710.10903.

[12] J. Tao, H. Wang, and T. Xiong, "2018 Selective Graph Attention Networks for Account Takeover Detection," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 49–54. doi: 10.1109/ICDMW.2018.00015.

[13] D. Cheng, X. Wang, Y. Zhang, and L. Zhang, "2022 Graph Neural Network for Fraud Detection via Spatial-Temporal Attention," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3800–3813, Aug. 2022, doi: 10.1109/TKDE.2020.3025588.

[14] N. Jiang, F. Duan, H. Chen, W. Huang, and X. Liu, "MAFI: GNN-Based Multiple Aggregators and Feature Interactions Network for Fraud Detection Over Heterogeneous Graph," *IEEE Trans. Big Data*, vol. 8, no. 4, pp. 905–919, Aug. 2022, doi: 10.1109/TBDATA.2021.3132672.

[15] D. Wang *et al.*, "A Semi-Supervised Graph Attentive Network for Financial Fraud Detection," in *2019 IEEE International Conference on Data Mining (ICDM)*, Beijing, China: IEEE, Nov. 2019, pp. 598–607. doi: 10.1109/ICDM.2019.00070.

[16] C. Lou, Y. Wang, J. Li, Y. Qian, and X. Li, "Graph neural network for fraud detection via context encoding and adaptive aggregation," *Expert Syst. Appl.*, vol. 261, p. 125473, Feb. 2025, doi: 10.1016/j.eswa.2024.125473.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[18] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language Modeling with Gated Convolutional Networks," in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 933–941. Accessed: Apr. 16, 2025. [Online]. Available: https://proceedings.mlr.press/v70/dauphin17a.html

[19] X. Bresson and T. Laurent, “Residual Gated Graph ConvNets,” Apr. 24, 2018, *arXiv*: arXiv:1711.07553. doi: 10.48550/arXiv.1711.07553.
[20] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” Feb. 22, 2017, *arXiv*: arXiv:1609.02907. Accessed: May 24, 2023. [Online]. Available: http://arxiv.org/abs/1609.02907
[21] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” Sep. 10, 2018, *arXiv*: arXiv:1706.02216. Accessed: Jul. 29, 2024. [Online]. Available: http://arxiv.org/abs/1706.02216
[22] W. Cong *et al.*, “Do We Really Need Complicated Model Architectures For Temporal Networks?,” Feb. 22, 2023, *arXiv*: arXiv:2302.11636. doi: 10.48550/arXiv.2302.11636.
[23] Y. Wang, H. Zhan, and W. Jiang, “Time Encoding Graph Attention Model for Financial Fraud Detection in Large-scale Financial Social Networks,” in *Proceedings of the 2024 3rd International Conference on Cryptography, Network Security and Communication Technology*, Harbin China: ACM, Jan. 2024, pp. 70–74. doi: 10.1145/3673277.3673290.
[24] Y. Kim, Y. Lee, M. Choe, S. Oh, and Y. Lee, “Temporal Graph Networks for Graph Anomaly Detection in Financial Networks,” Mar. 27, 2024, *arXiv*: arXiv:2404.00060. Accessed: Oct. 01, 2024. [Online]. Available: http://arxiv.org/abs/2404.00060
[25] K. Liu, H. Zhang, Z. Hu, F. Wang, and P. S. Yu, “Data Augmentation for Supervised Graph Outlier Detection via Latent Diffusion Models,” Nov. 23, 2024, *arXiv*: arXiv:2312.17679. doi: 10.48550/arXiv.2312.17679.