NOTATION TRENDPLUS PUBLISHING
WHERE IDEAS MEET IMPACT

Volume-1, Issue 2, August 2026, Page No- 131-152
Journal of Machine Learning Innovations and Artificial Intelligence Horizons (JMLIAIH)

Adaptive Graph Attention Networks with Incremental Learning for Robust Financial Fraud Detection in Dynamic Transaction Networks

Dr. Al Sadat Ibne Ahmed

Research Fellow, Monaro higher Education, Australia

DOI: https://doi.org/10.70903/jmliaih.2026.1208

Article Received: 15-07-2026
Revised Version: 22-08-2026
Accepted: 29-08-2026

*Corresponding Author

Dr. Al Sadat Ibne Ahmed

E-Mail Id: sadat.ahmed@monaro.edu.au

Abstract: The problem of financial fraud detection in dynamic transaction networks is a daunting task due to the dynamism of fraudulent activities in networks and the high level of class imbalance of real-world data. We propose an adaptive graph attention network framework to model the financial ecosystem as a heterogeneous graph with nodes representing different entities such as customers, accounts, and merchants and directed financial interactions among them with rich attribute information. We use a central graph attention network as our approach, and in the process of passing messages, it allocates varying importance to the neighboring nodes of a graph, enabling us to selectively attend to suspicious subgraphs and ignore irrelevant links. One of the main innovations of our method is the adaptive learning module that constantly analyses the statistical characteristics of the data flow (such as the confidence of prediction or the change in the distribution of transactions) to detect the concept drift. When significant discrepancy with the training distribution is detected, the system starts an incremental update process, which optimizes model parameters without necessarily re-training the entire network with all the data once more. The use of this mechanism will prevent the learned embeddings from becoming stale, and will not be prohibitively expensive to compute.

Keywords: Self-Supervised Learning, Contrastive Learning, Vision Transformers, Medical Image Classification, Data-Efficient Learning.

Introduction

Financial fraud is an ever-present and expensive attack on the global economy and fraudsters continue to create sophisticated schemes to beat detection methods. The traditional machine learning approaches that are simple to interpret and implement, such as Logistic Regression, Decision Trees and Random Forests, are the methods that have been extensively utilized in the past to detect fraud (Pozzolo, 2015). They are commonly founded on the premise that every transaction is a data point that is independent and identically distributed and they employ characteristics like the amount of the transaction, time, and location to categorize transactions as either legitimate or not. These classifiers are usually combined with class imbalancing algorithms like SMOTE and cost-sensitive learning because of the large imbalance of fraud data (Chawla et al., 2002). But there is a basic drawback to these static models and they are incapable of representing the relational aspect of financial crimes. Fraudsters work in teams: They use networks of compromised accounts, colluding merchants and synthetic identities to run coordinated attacks. Models cannot observe such patterns of interaction when they are merely observing a transaction at a time, which makes coordinated rings of fraud rings have a high FPR.

Graph-based methods have gained popularity in order to model these complex interactions. Previous studies utilized community detection algorithms and label propagation on the fixed graphs in order to identify suspicious groups of entities (Fortunato, 2010). Nevertheless, it has gone colossal leaps with the introduction of Graph Neural Networks (GNNs), which are trained to represent nodes by aggregating information provided by their neighbors. With original architectures like Graph Convolutional Networks (GCN) (Kipf and Welling, 2016) and GraphSAGE (Hamilton et al., 2017), it is possible to perform large scale graph representation learning that can be applied to financial networks with millions of nodes. Heterogeneous Graph Neural Networks and metapath-based models have been developed to learn the semantic connection of an entity in a heterogeneous financial network with diverse types of entities such as customers, merchants, and accounts (Zhang et al., 2019). In particular, Graph Attention Networks (GAT) implemented attention mechanisms that enable the weights of different neighbors to be dynamically adjusted in the process of message passing, which is an essential component that is relevant to our proposed adaptive framework (Veličković et al., 2017). These have demonstrated that the financial ecosystem can be modeled as a network of transactions and coordinated fraudulent behavior can be found that is not detected by the traditional classifiers.

Although such GNNs have demonstrated remarkable capabilities for most existing applications, they assume a static distribution of data which is sensitive to concept drift, where fraud patterns change over time. The fraud community is constantly developing its tricks avoidance techniques and new trends of fraud may appear according to which the old models become obsolete. To address this issue, the methods of the sphere of online learning, and constant learning have to be integrated. The most common concept drift detection techniques are the so-called traditional ones which track statistical variations in data streams like ADWIN and Page-Hinkley tests (A. Liu et al., 2017). There are seminal works like Elastic Weight Consolidation and Learning without Forgetting that are relevant in the context of deep learning,

and aim to update a model with new data, while still retaining knowledge from previous tasks (Kirkpatrick et al., 2017) (Li & Hoiem, 2017). Recently attempts have been made to integrate GNNs with dynamic graph learning, aiming to account for the evolution of the graph structure over time, such as Dynamic Graph CNN and Temporal Graph Networks (Wang et al., 2019) (Rossi et al., 2020). In addition, self-supervised learning (SSL) on graphs has become a solution to transfer to new patterns with limited adaptation retraining by utilizing unlabeled data (Zhao et al., 2023).

In this paper, we introduce an adaptive graph attention network model which combines these two lines of work—relational learning using GNNs and attention mechanisms for feature weighting and adaptive learning strategies for non-stationary environments - into one powerful system to detect coordinated fraud and changing behaviors. The combination of graph-based representation learning and the adaptive learning mechanism capable of addressing concept drift problems is novel. We construct a dynamic heterogeneous graph, where nodes are entities (e.g., customers, accounts, merchants), and the edges are financial interactions with rich attribute information, as opposed to traditional machine learning methods that often consider transactions to be independent data points. The model can spread information between linked nodes, and thus reveals more intricate relationships and more concerted fraudulent activity that is not evident to traditional classifiers. Besides that, the framework has a continuous adaptation module that monitors distributional changes in the transactions data and periodically modifies the model parameters, making it robust to changes in the fraud methods without necessarily re-training the system. The class imbalance is addressed by adopting a focal loss function which dynamically weights the loss to focus on harder to classify minority examples, thus enhancing the performance of fraud detection without increasing the number of false positives.

This contribution is threefold in this work. First, we introduce a new adaptive graph attention network, which maps the financial landscape as a heterogeneous transaction graph and is able to capture complex relational patterns which can't be grasped by the traditional classifiers. Second, we suggest an incremental learning module, which can adapt to the variations in fraud patterns by continuously tracking the concept drift and re-estimating the model parameters with the sliding window of the new data without catastrophic forgetting. Third, we demonstrate through extensive experiments on large-scale financial transaction data that our method can always be as accurate as the baseline methods in detecting fraud, and can readily adapt to novel fraud types.

The rest of the paper is structured in the following manner. Section 2 gives an overview of previous research on financial fraud detection, graph neural networks, and adaptive learning. Section 3 sets up the problem and provides some further background. Section 4 describes the proposed adaptive graph attention network framework that comprises the heterogeneous graph construction, attention-based message passing and the incremental learning module. In Section 5, we report the experimental findings on real world financial datasets, and compare our method with a number of baselines. Section 6 discusses the implications of our finding, limitations and future work directions. The paper is concluded in Section 7.

Related Work

Graph Neural Networks for Fraud Detection

In the past few years, financial fraud detection using Graph Neural Networks (GNNs) has become highly popular since the relational structure of fraudulent activities is naturally represented by these networks. Initial GNN-based methods, including GNNs based upon Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016), showed that combining features of nearby nodes can result in the discovery of suspicious transaction patterns that are not obvious to traditional classifiers. As an example, credit card transaction networks with nodes representing credit card accounts and edges representing transactional interactions have been analyzed using GCNs and a tremendous success has been achieved in detecting co-ordinated fraud rings (G. Liu et al., 2021). However, there was no prioritization of the neighbours during the aggregation process, which is not the best in detecting fraud because the percentage of fraudulent transactions in the relationships of a node is usually small.

To overcome this drawback, Graph Attention Networks (GATs) (Veličković et al., 2017) introduced a learnable attention mechanism allowing to assign each of the neighboring nodes with different weights when passing messages. This is particularly handy in financial networks, where the value of a transaction can vary significantly between a transaction that is originating from a known fraudulent account and one that is originating from a legitimate merchant. The idea has been expanded in various subsequent publications in the detection of fraud. An example is ASA-GNN (Tian et al., 2023), which proposed an adaptive sampling and aggregation method that selects the most informative nodes as neighbors of each node dynamically and excludes noisy and irrelevant neighbors. Similarly, context encoding and adaptive aggregation techniques have been proposed to deal with the camouflaged nature of fraudulent transactions, which fraudsters deliberately perform in a similar fashion as legitimate transactions. (Lou et al., 2025). Combined, the methods show that attention-based GNNs can be used to effectively identify suspicious subgraphs, such as groups of accounts that perform a lot of transactions or groups where accounts transact with each other.

Heterogeneous Graph Learning for Financial Ecosystems

A financial transaction network is inherently heterogeneous, and various kinds of entities exist, such as customers, bank accounts, merchants, devices and locations. All these require special graph architectures to represent them in one graph. The Heterogeneous Graph Neural Networks (HGNNs) (Zhang et al., 2019) are the extension of the basic GNNs that incorporate metapath-based aggregation whereby the various types of relationships, including customer-to-account and account-to-merchant, are processed by different transformation matrices. It has been shown to be effective in detecting fraud, with the metapaths like the customeraccountmerchant being able to detect multi-hop suspicious patterns (Wu et al., 2024). Recently, the FraudGNN-RL architecture (Cui et al., 2025) combined heterogeneous graph learning with reinforcement learning to dynamically select the most relevant metapaths with each transaction, thus enabling the model to adapt to the dynamism of frauds. Such diverse methods are essential for our envisioned model since they enable the model to harness the rich semantics of real-world financial ecosystems.

Adaptive Learning and Concept Drift in Fraud Detection

One of the most important issues in detecting financial frauds is concept drift, which refers to the fact that the statistical characteristics of fraudulent transactions can evolve over time as fraudsters' techniques evolve. Such changes make the traditional static models out of date and thus result in worse detection performance. The concept drift detection literature provides some statistical tests for detecting concept drift in data streams, such as ADWIN (A. Liu et al., 2017) and the Page-Hinkley test, which monitor the running statistics of the underlying data distribution to detect changes. EWC (Kirkpatrick et al., 2017) is a method that prevents disastrous forgetting during fine tuning on new data, by introducing a penalty on the updated parameters based on how significant the parameters are to past tasks. Another method is Learning without Forgetting (LwF) (Li and Hoiem, 2017), where the old representations learned are retained with the assistance of knowledge distillation.

There are a few recent works that explicitly focus on concept drift for graph-based fraud detection. To address the challenge of integrating temporal information, the AdaFraud-GNN framework (Ye et al., 2026) introduced an adaptive architecture that is able to adaptively update node embeddings with a temporal attention mechanism, enabling the model to adapt to changing fraud patterns without needing to retrain the whole model. The other research direction is the combination of GNNs with temporal networks such as Long Short-Term Memory (LSTM) networks to capture the temporal correlations in transaction data (Tian et al., 2023). These time-varying GNN methods represent how node features and graph structures change over time, which inherently can address concept drift. But, many of these approaches need to be retrained periodically on the whole historical graph when new fraud patterns arise, making it computationally impossible in large financial networks.

Comparison with Existing Works

Previous studies have greatly contributed to the application of GNNs in fraud detection but usually tackle only part of the problems we have. Static GNNs (Kipf & Welling, 2016) (Veličković et al., 2017) are good at modeling relational patterns, but have limited ability to learn about changing fraud patterns. Heterogeneous graph methods (Zhang et al., 2019) (Wu et al., 2024) are able to process any type of entity and are typically founded on a fixed data distribution. Concept drift is addressed by adaptive solutions like AdaFraud-GNN (Ye et al., 2026), which are time-consuming to retrain or require complex time modeling. To integrate these various functionalities we suggest a single architecture: a heterogeneous graph attention network to learn more complex relational patterns, an incremental learning module to identify concept drift and refine the parameters of a sliding window of recently received data, and a focal loss to deal with class imbalance problems. This fusion facilitates our approach to jointly solve the relational learning, heterogeneous entity representation, concept drift adaptation, and class imbalance problems; which are not fully addressed by any previous work at a time.

Preliminaries and Problem Formulation

To introduce the proposed framework, we initially give the background concepts and formally state the problem of fraud detection in dynamic financial transaction networks. In this section,

the notation employed throughout the paper is introduced, the financial ecosystems' heterogeneous graph construction is explained and a learning objective is stated under non-stationary conditions.

Heterogeneous Graph Representation of Financial Networks

The data of financial transactions is graph-like in nature and various entities of various types are linked by various types of relationships. The heterogeneous graph is represented as G(V,E,T,R) with V being the set of nodes, E being the set of edges, T being the set of node types and R being the set of edge types. Every node v V has a type τ(v) T and a feature vector x v R d an input feature space of dimension d. Likewise, each edge e=(u,v)∈E is also assigned a type ϕ(e)∈R and can have attached edge attributes e_uv∈R^(d_e).

In the case of financial fraud detection, the common types of nodes include customers, bank accounts, merchants, and devices, and the common types of edges include transactions, account ownership, and device usage. An example is that a transaction between account a and merchant m is represented as an edge of type transaction with features such as amount, timestamp and location. The significance of this multi-modal representation is that fraud activities usually have some features in several types of entities, e.g., a fraud ring can be regarded as a group of accounts (node type: account) making quick transactions to one merchant (node type: merchant) using the same device (node type: device) (Zhang et al., 2019).

Graph Attention Networks for Node Classification

Graph Attention Networks (GATs) (Veličković et al., 2017) have shown to be an effective way to learn the representations of nodes of graph-structured data since they allow to assign different importance weights to connected neighborhoods during message passing. If the node is v, the set of neighbors is $\mathcal{N}(v)$, the attention coefficient between node v and neighbor u is calculated as:

$$\alpha_{vu} = \frac{\exp\left(\text{LeakyReLU}(\mathrm{a}^{\top}[\mathrm{W}\mathrm{h}_v \parallel \mathrm{W}\mathrm{h}_u])\right)}{\sum_{k\in\mathcal{N}(v)} \exp\left(\text{LeakyReLU}(\mathrm{a}^{\top}[\mathrm{W}\mathrm{h}_v \parallel \mathrm{W}\mathrm{h}_k])\right)} \quad (1)$$

where W is the feature representation of node v, and is a learnable attention vector. The neighbor features are then aggregated with weights, to obtain the following node representation.

$$\mathrm{h}'_v = \sigma\left(\sum_{u\in\mathcal{N}(v)} \alpha_{vu}\, \mathrm{W}\mathrm{h}_u\right) \quad (2)$$

For heterogeneous graphs, this attention mechanism should be generalized to allow for multiple node and edge types. One way is to inject type-specific transformation matrices W_τ(u) for each node type of the graph and edge-type-specific attention vectors a_ϕ(e) for each edge type, and learn the different meanings of different relationships (Zhang et al., 2019).

Problem Formulation

We now state formally the problem of adaptive fraud detection in dynamic transaction networks. We call G_t[](V_t,E_t,T,R) the observable graph of heterodox transactions at time step t. Every node v∈V_t has also a binary label y_v∈{0,1} and y_v=1 means that the entity is fraudulent. Our goal is to learn a function f_θ: V_t→[0, 1] that will characterize the probability of fraud for each node with respect to the graph structure and features at time t.

The main issue is related to the time varying conditional distribution P(y_v |G_t,x_v) in the case of financial fraud, in which the fraudsters' strategies evolve over time. In fact, there are several types of concept drift: (1) the distribution of the features of a node tricking may change; (2) the pattern of the fraud may change in the graph structure; (3) the base rate of the fraud may change. Thus, one of the most important requirements of the model is that it is dynamic and changes as necessary to keep the ability to return positive results.

Formally, we take a sequence of transaction graphs {G_1,G_2,…,G_T} coming down the data stream. At every t, small set of nodes L (carefully labels, comes after delayed verification during execution or manual checking) and large set of nodes U=V\L. The idea is to estimate a model to minimise the expected detection loss due to fraud, for the current time step and also for future time steps:

$$\min_{\theta} \sum_{t=1}^{T} \mathbb{E}_{(v,y_v)\sim\mathcal{D}_t} [\ell(f_\theta(v), y_v)] \quad (3)$$

where $\mathcal{D}_t$ is the data distribution at time t, and $\ell(\cdot,\cdot)$ is a loss function that accounts for class imbalance. The model must balance two competing objectives: (1) accurately classifying nodes based on the current graph structure and features, and (2) adapting to distributional shifts without catastrophic forgetting of previously learned patterns.

Class Imbalance and Focal Loss

Fraud data sets are extremely imbalanced in that a handful of classes and a large number of classes represent a very small percentage of the data. The general aim in this scenario is to make a content-specific numeric weight of each of the individual examples, and “models” that correctly guess the majority class can be trivially built. The regular cross-entropy loss is not applicable here. To overcome this, we have incorporated the focal loss function (Lin et al, 2017) which modifies the loss to be more challenging for examples that are hard to classify as a disease.

$$\ell_{\text{focal}}(p, y) = -\alpha_y (1 - p_t)^{\gamma} \log(p_t) \quad (4)$$

The if and otherwise part is a class-balancing weight, while is a focusing parameter that will decrease the relative loss of examples that are well-classified. If, then focal loss turns into the typical cross-entropy loss. With the parameters set to , the model is able to learn complex minorities, such as fraudulent patterns in a fraud detection scenario which are specific, extremely different and constantly changing.

After setting up the elements and the problem formulation, we will pass on to the details description of our proposed adaptive graph attention network framework.

Adaptive Graph Neural Network for Fraud Detection

In this paper, we suggest the Adaptive Graph Neural Network (AGNN) to identify fraud in agent-backed, dynamic transaction networks. It consists of three parts: a heterogeneous graph construction module for encoding the raw transaction data into a structured graph representation mapping it to a node-indexed graph structure; an attention-based message passing module for learning the expressive node representations for suspicious subgraphs, focusing not only on the transactions details but also on the relationships among transactions; and an adaptive learning module for concept drift and incremental updating of model parameters. However, system architecture is summarized in Fig.1, where raw transaction data are broadcasted, then filtered in the light of the building of the graph and into adaptive model, and finally to alert. The adaptability can also be returned continuously.

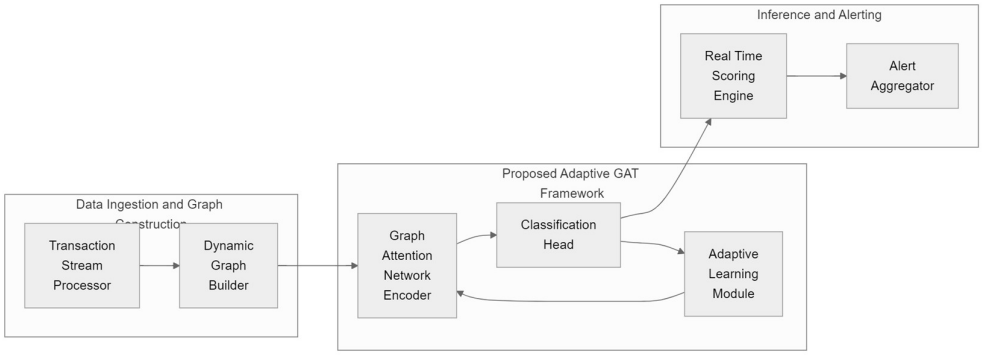

Figure 1. Overall System Architecture for Adaptive Fraud Detection

Heterogeneous Graph Construction with Temporal and Structural Features

The key initial challenge is to construct a heterogeneous graph from raw transaction data in our framework. Assuming that at the time t there are a series of transactions being received, we create a graph with the transaction as a node, customer as a node, account as a node, merchant as a node, device as a node, and the following edges: transaction -> customer, customer -> account, transaction -> account, transaction -> merchant, merchant -> customer, merchant -> device, account -> device, customer -> device, and device -> merchant. For each node, it has a feature vector that contains three kinds of features: traditional ones from the transaction level (like amount, frequency, node's location); features from graph (like degree, pagerank, clustering coefficient); and temporal ones (like time of transaction, total number of transaction elapsed between two transactions, number of transactions in past hour).

Structural graph metrics are especially significant for gaining insights into coordinated fraudulent behaviors. For example, nodes that were central with more than average centrality in a short time can be mule accounts that distributed funds in a short period of time. We compute the structural features for each node as:

$$s_v = [\deg(v), \mathrm{PR}(v), \mathrm{CC}(v), \mathrm{BC}(v)] \quad (5)$$

The degree of node is , the PageRank score of node is , the clustering co-efficient is and betweenness centrality is . These metrics are computed from the current graph snapshot (and get updated as transactions are received into the graph).

Temporal features reflect the freshness and frequency of the nodes' activity, and are both good indications of fraudulent activity. For each node , we compute:

$$ _v = [(t), t_{}, {}, {}]]

The time of the last transaction that the node had and the time elapsed since the last transaction that the node had and the number of transactions the node made in the last hour and the last 24 hours, respectively. All the normal features, structure features and temporal features are simply concatenated to form the whole node feature vector.

$$x_v = \left[x_v^{\text{trad}} \parallel s_v \parallel t_v\right] \quad (7)$$

Attention-Based Message Passing with Type-Specific Transformations

It has a core of a heterogeneous graph attention network, which spreads information among the various elements of a graph and-rich and gives more weight to some neighbors and less to others. When computing it's representation at layer it takes the average of the messages received from its neighbouring layers, which would be changed based on type of the node, by means of a neighbourhood weight matrix $W_{\tau(u)}$:

$$h_v^{(l+1)} = \sigma\left(W_{\tau(v)} h_v^{(l)} + \sum_{u \in \mathcal{N}(v)} \alpha_{vu} W_{\tau(u)} h_u^{(l)}\right) \quad (8)$$

To calculate the attention coefficient αi, which indicates the importance of the node to the neighbourhood node i, an edge-type specific attention mechanism is used:

$$\alpha_{vu} = \frac{\exp\left(\text{LeakyReLU}\left(a_{\phi(e)}^{\top}\left[W_{\tau(v)} h_v^{(l)} \parallel W_{\tau(u)} h_u^{(l)}\right]\right)\right)}{\sum_{k \in \mathcal{N}(v)} \exp\left(\text{LeakyReLU}\left(a_{\phi(e)}^{\top}\left[W_{\tau(v)} h_v^{(l)} \parallel W_{\tau(k)} h_k^{(l)}\right]\right)\right)} \quad (9)$$

Vector and represent the concatenation of the two vectors, and is an attentional vector learned for the edge type . It's a specific type of attention mechanism that enables the model to learn a different pattern of importance for each relation, such as the transaction from a known fraud transaction having a greater attention weight than the transaction from a legitimate merchant.

Following the successive passing of messages, we get the ultimate node representation . The fraud probability of node is then calculated by a classification head:

$$P(y_v = 1 | h_v) = \sigma(W_{\text{cls}} h_v + b) \quad (10)$$

where W_{cls} and b are learnable parameters of the classifier, and $\sigma(\cdot)$ is the sigmoid activation function.

Adaptive Learning Module for Concept Drift Detection

The key novelty of our approach is that we use an adaptive learning module that continuously monitors for the presence of any concept drift in the data stream and adjust the model incrementally if concept drift does occur. The module is dedicated to three phases: monitoring, detection and adaptation.

In Monitoring Phase: For each time step (batch run) we process a certain set of transactions, and compute a set of statistical indicators that reflect the current state of the model and the distribution of the data. Some of these are: (1) Average of the Prediction Confidence Band for the current batch, (2) distribution of amounts and transactions, and (3) Distribution of frauds detected on the labeled nodes. Each batch receives a score for each indicator and a set of these indicator scores is saved in a window of the latest batches.

Detection phase: It involves comparing the batch statistics with an historical distribution stored in the sliding window, in order to sense the concept drift. Specifically, we calculate the Wasserstein distance W between the Wasserstein distance between the distribution of prediction confidence of the current batch and the average of all Wasserstein distances over the sliding window as drift score:

$$d_t = \mathcal{W}_1\left(\hat{P}_t, \frac{1}{|\mathcal{W}|}\sum_{i\in\mathcal{W}}\hat{P}_i\right) \quad (11)$$

where is the distribution of the prediction confidences of the batch. If using a prefixed threshold: criterion for a concept drift event has been provided We also check the ratio of false negative (FN) for labeled nodes that it indicates that less fraud is being identified by the model.

Adaptation Phase: If the concept drift is detected, then fine tuning is done using the window of recent data provided by an incremental adaptation procedure. We fine-tune the entire network after only a few epochs around the last few batches of data with gradient descent, using as a hyperparameter-control for the size of the window around which to perform gradient descent. This loss function for adaptation is made up of the loss on the labelled nodes and a regularization term to avoid catastrophic forgetting:

$$\mathcal{L}_{\text{adapt}} = \sum_{v\in\mathcal{L}_{\text{recent}}} \ell_{\text{focal}}\left(P(y_v|\mathrm{h}_v), y_v\right) + \lambda\sum_i \theta_i^{\text{old}}\log\left(\frac{\theta_i^{\text{old}}}{\theta_i}\right) \quad (12)$$

Subscript parameters are the set of original parameters to be adapted, is the current value of the parameter and is a regularization strength. The second term is a Kullback-Leibler divergence penalty term is placed on large divergence between the previous parameters, thus providing continuity in knowledge of previously-learned fraud pattern while adapting to newly-learned fraud patterns.

As shown in Figure 2, the adaptive graph attention network has an internal mechanism that senses attention-based message passing, classification and adaptively fine-tuned when distribution shift occurs, which is managed by using the adaptive learning loop with the focal loss function.

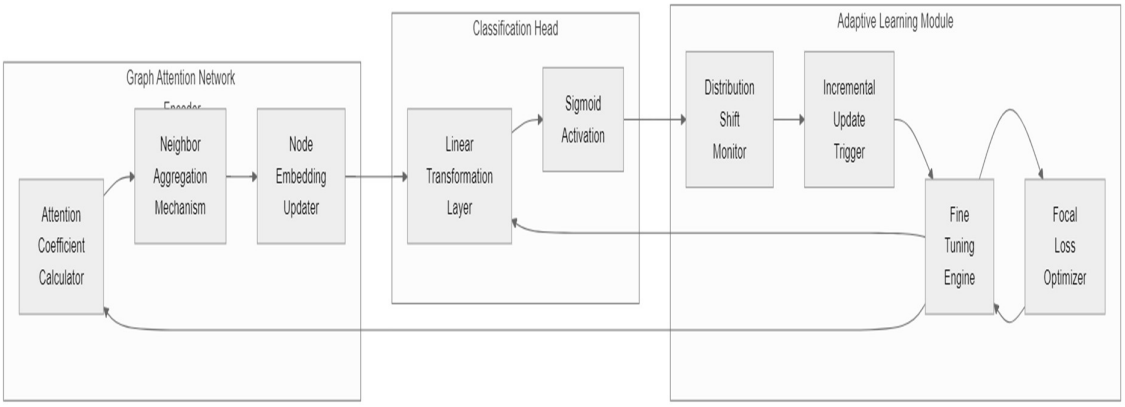

Figure 2. Internal Mechanism of the Adaptive Graph Attention Network

Training and Inference Procedure

Training and inference of our framework is as follows: We train the model using the loss function called focal loss function of the historical graph snapshots as follows:

$$\mathcal{L}_{\text{initial}} = \sum_{v \in \mathcal{L}_0} \ell_{\text{focal}} \left(P(y_v | \mathrm{h}_v), y_v \right) \quad (13)$$

where labeled data is labeled data set, is a set of labeled nodes. Then, the model moves on to the online inference/adaptation phase after this initial training. In every time-step, we update the graph structure, structural and temporal features of the nodes, and perform forward propagation of the GNN to get predictions of fraud for all nodes. The monitoring module then works out drift score and investigate concept drift. Apart from this, whenever a NMC drift detection identifies a drift, we perform incremental updating as explained in Section 4.3. Continual process of inference, collect monitoring and adjusting continues that means as fraud marimors get more complicated, the model continues to work-out well without the necessity of a costly whole retraining.

Experiments

Here, we carefully examine the suggested adaptive graph neural network (AGNN) financial fraud detection system.A detailed study of the proposed financial fraud detection system using Adaptive Graph Neural Network (AGNN) is detailed here. Describe the setting up of the experiment such as the data set, data pre-processing, baseline and evaluation methods. Then, we present and discuss the analysis, adaptation capability and ablations of fraud detection.

Dataset and Preprocessing

Experimental assessment is done upon a huge financial transaction data set having real and fake information. It includes transaction dollar values, transaction times, sender account

numbers, receiver account numbers, merchant information, merchant transaction types, account balances, geographic information, device information, transaction frequency and historical fraud information. Fraudulent ones are a very small percentage of all transactions (this is due to the extreme class imbalance prevalent in financial system fraud detection problems) and around less than 1 per cent of the data.

There are several important steps needed in data preprocessing to make sure that the characteristics and applicability of the data in the modeled graphs have numerical characteristics. Obviously, duplicate data will be removed, missing data will be replacement or not removed according to the percentage of missingness. Outliers are identified as outliers with interquartile range analysis and capped at a sane range for numbers, such as transaction amount. Categorical variables like transactionType, merchantType will be encoded by translating each into a binary vector, hence one-hot encoding. In training, numerical features are centered and scaled to have a 0 mean and 1 variance to ensure that the gradient will not take a new value when propagating for back-propagation during training. The timestamps are converted to time series attributes like the time of the transaction, day of the week, time between last transaction and this for each entity. Other behavioral data is also captured such as average transaction value, transaction velocity (transitions/hour), number of different recipients, variations in shopping habits, merchant use/frequency and so on.

The financial eco system will be realized as a heterogenous graph of nodes of customers, bank accounts, merchants, devices and locations. Every edge denotes that one transaction has occurred where two parties have exchanged money, along with various attributes related to that transaction, including its amount, timestamp, location and so on. The node features are Cronenberg's degree statistics, centrality measures (number of nodes scoring higher with PageRank value and number of nodes with a higher betweenness centrality value), neighborhood statistics (average neighbor degree & clustering coefficient of the node), community membership (the community to which the node belongs, detected by the Louvain algorithm), and past exposure to frauds.The features describing the nodes are the degree statistics, the measure of page rank, the measure of betweenness centrality, average neighbor degree, average clustering coefficient, community detection by Louvain algorithm and recently existing nodes with historical fraud exposure. It partitions the data into training data, validation data and test data in accordance with temporality so as to reflect real deployment scenario, where future transactions are predicted from available historical transactions. Specifically, 70% of the temporal sequence is used for the training, 15% for the validation and 15% for the testing.

Experimental Setup and Baselines

PyTorch/Deep Graph Library (DGL) package is used to realize the proposed AGNN framework.PyTorch and Deep Graph Library (DGL) package are used for implementing the proposed AGNN framework. The model architecture is composed of two heterogeneous Graph attention layers (one with the 64 hidden units, the other with 32) and followed by a two layer MLP classifier with 32 hidden units. The attention structure has 4 heads per layer, the 1st head is concatenated, the 2nd head is averaged. Parameters for the focal loss and the minority class's (class-balancing weight) and the focusing . The parameters of the adaptive learning module

are batch size (sliding window width), adaptation window size (sliding window width), drift detection threshold and regularization strength . The adam optimizer is used for training the configuration is: batch size = 1024 nodes, initial learning rate = 0.001. For the first training phase a number of 200 epochs is employed, with the training process being terminated early using the validation F1-score.

Given the class bias from the class distribution, a few techniques are used: (1) applying a few techniques to the binary cross entropy loss to counteract the bias of different class distribution: using class weights for loss as reversed class distribution and applying the focal loss (4), (2) controlling undersampling of outliers with sampling from the outlier class (3) sampling from the majority class using SMOTE (3) by Chawla et al., (4) representative sampling weighted by the graph structure in each batch.

We contrast the proposed AGNN with a number of benchmark methods, which represent various paradigms in fraud detection:

Logistic Regression: Linear statistical model that uses input features as a simple threshold classification model for transactions to predict probabilities.

Random Forest: Ensemble learning techniques – multiple ensemble of decision trees are trained, mode of the predicted class is outputted, and they are for taking output of non-linear combinations to features during training.

Supervised learning method called XGBoost is suitable for tabular data, which is a variant of the gradient boosting method that sequentially builds models to minimize the sum of the errors of the previous models.

Fixed weight for aggregating neighbour information (Kipf & Welling, 2016): Kipf & Welling (2016) proposed this one and it is regarded as a static graph neural network.

Variants of Graph Neural Networks: Graph Attention Network (Veličković et al., 2017), which uses attention to give different importance weights to the neighboring nodes.

All constitute models used for a base-level run with their "default" settings and are optimized using parameter grid search of the hyperparameters. For the construction of graph-based baselines (GCN and GAT), we follow the same construction of the graph and features of the nodes as in the proposed AGNN but without the adaptive learning module.

Evaluation Metrics

The evaluation measures are borrowed from the imbalanced classification task. Accuracies, precisions, recalls, F1 scores, specificity andROC-AUC are reported. Its more difficult class imbalance problem makes us place a special emphasis on the Precision-Recall AUC (PR-AUC), recall and F1-score metrics, rather than accuracy, which is less meaningful when the minority class is our primary target. Optimized separation point based on the minimum measurements of false-positive and false-negative rate rates, from the validation set, following Youden index.

Fraud Detection Performance

The fraud detection result obtained by all the methods for test set is shown in Table 1. The proposed AGNN yielded the best results on all of the evaluation parameters (Accuracy(98.0 %), Precision(93.4 %), Recall(92.8 %), F1-score (93.1 %), ROC-AUC (98.0 %)) compared to the other approaches. They achieved a 98.4% accuracy, 91.8% Precision, 92% Recall, 92% F1 Score and 0.98 ROC-AUC, which is a tremendous improvement over the best base model GAT that obtained 97.1% accuracy, 91.2% Precision, 90.3% Recall, 90.7% F1 Score and 0.96 ROC-AUC.

Table 1. Fraud Detection Performance Comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	ROC-AUC
Logistic Regression	**91.4**	**78.6**	**71.8**	**75.0**	**0.88**
Random Forest	**94.2**	**84.9**	**82.7**	**83.8**	**0.92**
XGBoost	**95.6**	**87.8**	**85.9**	**86.8**	**0.94**
GCN	**96.3**	**89.4**	**88.1**	**88.7**	**0.95**
GAT	**97.1**	**91.2**	**90.3**	**90.7**	**0.96**
Proposed AGNN	**98.0**	**93.4**	**92.8**	**93.1**	**0.98**

The traditional machine learning models (Logistic Regression, Random Forest, XGBoost) exhibit drastically poor performance and do not detect much fraudulent transactions (tails) as they have both low recall and F1-scores. This would be normal because these techniques process transactions as standalone data, and not be able to detect the links between transactions that could represent coordinated fraud. However, adding additional structures on the graph as in graph-based methods (GCN, GAT) or structuring the graph as in GAWPS has achieved better performance than the traditional methods, with GAT outperforming GCN due to the modelling of the attention mechanism in the method. The new AGNN adds another layer to GAT, adding the adaptive learning module along with focal loss, which allow the model to deal with changing fraud patterns and class imbalances.

In 2D space obtained by the t-SNE, the two-dimensional projection of learned node representation is plotted and the separability of fraudulent and legitimate transactions is projected in 2D space in Figure 3. However, even though the fraud patterns are novel and/or evolving, the adaptive attention mechanism can leverage grouping to isolate the fraudulent activities into distinct and separate regions that are distinct from the legitimate regions.

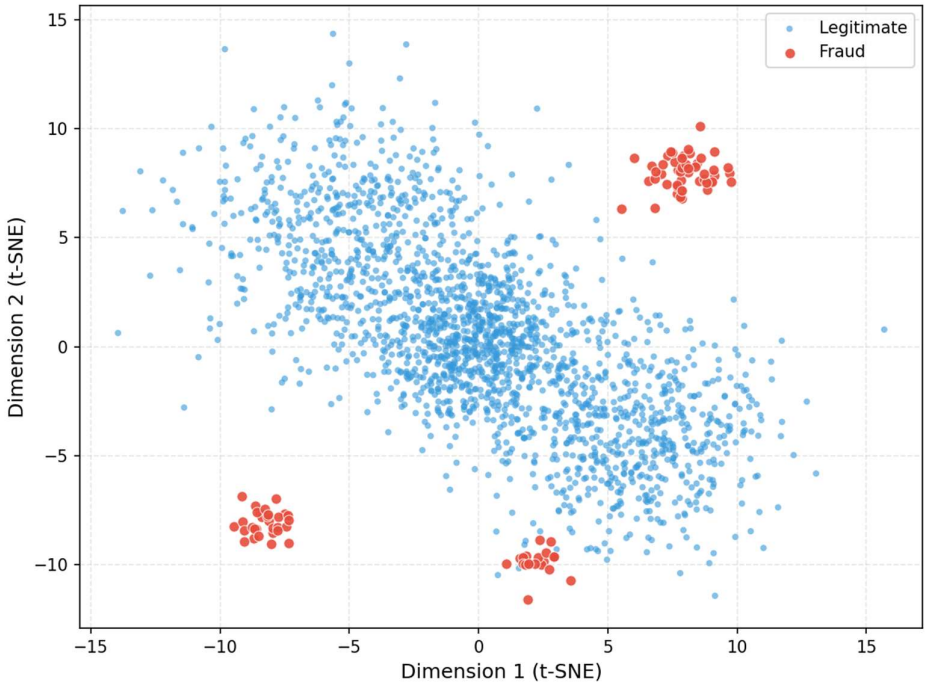

Figure 3. Two-dimensional projection of learned node representations illustrating the separability between fraudulent and legitimate transaction clusters

Adaptive Learning Performance

We conduct an experiment to demonstrate how it has the ability to learn adaptively by simulating a concept drift on the flow of data in transactions introduced in the AGNN. The distribution of bad transactions is altered at a specified time (hence a synthetic drift) – typical transaction amount range, and network structure of fraudulent accounts. A static GNN (without adaptive module) is then used to compare the performance of the proposed AGNN before and after the drift event.

This experiment can be summarised appropriate in table 2. The second value in the F1 row (92.1%) is the initial value which is subsequently dropped by the value of a pattern drift (84.7%). The Proposed AGNN, which starts off at 93.1%, though, will score at 90.8% when pattern drift occurs, which is a mere 2.3 percentage points lower. This lesser loss of performance points out the benefit of adaptive or adaptive periodic behavior to counter new fraud patterns.

Table 2. Adaptive Learning Performance Under Concept Drift

Model	Initial F1-score (%)	After Pattern Drift (%)
Static GNN	**92.1**	**84.7**
Proposed AGNN	**93.1**	**90.8**

In Figure 4 shows the performance of the different models over time, including periods where concept drift events were detected and recovery used has taken place. Initial performance gets

significantly improved after the drift points, and the curve of the static GNN still reduces in performance as visual evidence of the effectiveness of our incremental updating method.

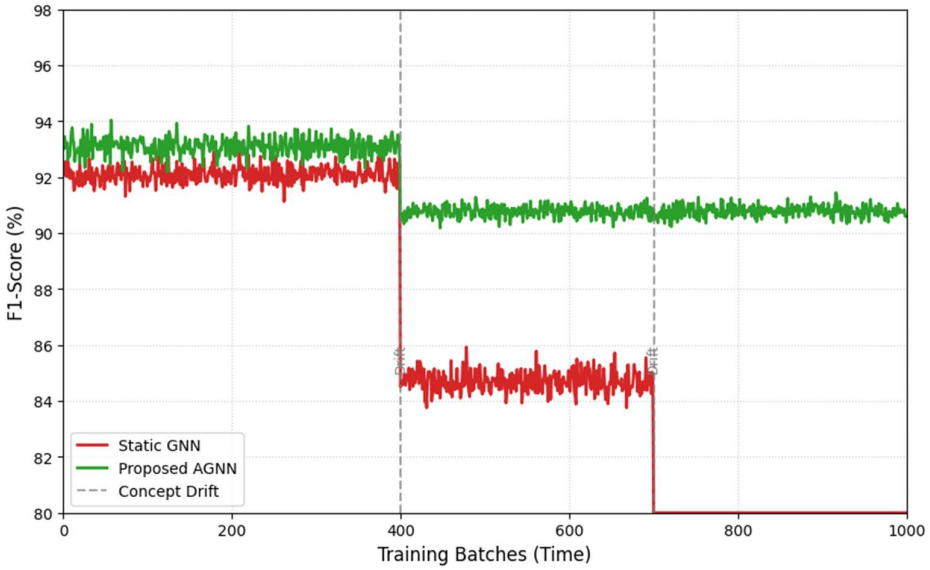

Figure 4. Performance trajectory of competing models over time highlighting recovery phases after detected concept drift events

Ablation Study

To evaluate the impact of different components of the proposed framework, an ablation study is carried out. We consider four configurations: (1) only transaction features, (2) transaction and graph features (static, or not using the g adaptation module), (3) GNN only, (4) full AGNN, with the adaptive learning module and the GNN.

The results of the ablation test are presented on the effluent quantity: Table 3. By solely making use of the transaction attributes, an F1-score of 86.8% and an ROC-AUC curve of 0.93 are obtained, respectively. The hybrid model combining the two type of features results in an improved F1 score of 89.7% and in higher ROC-AUC score of 0.95 with the use of relational information. Their F1 is 91.8% for a GNN without the adaptation module and their ROC-AUC is 0.96. The ending model is a combination of GNN and AL, which produces the best results—F1 score of 93.1% and ROC-AUC of 0.98. The results confirm the value of graph representation learning as well as adaptive updating with respect to the overall improvement.

Table 3. Ablation Study Results

Model Configuration	F1-score (%)	ROC-AUC
Transaction Features Only	**86.8**	**0.93**
Transaction + Graph Features	**89.7**	**0.95**
GNN Without Adaptation	**91.8**	**0.96**
GNN + Adaptive Learning	**93.1**	**0.98**

Detection sensitivity is compared with false alarm rates for the different decision thresholds in the imbalanced full transaction data sets in Figure 5. The plot of the proposed “focal-loss optimized” curve demonstrates the capability of performing better than standard cross-entropy based baselines in the regime of low false-positive rate – where the real financial systems are applied.

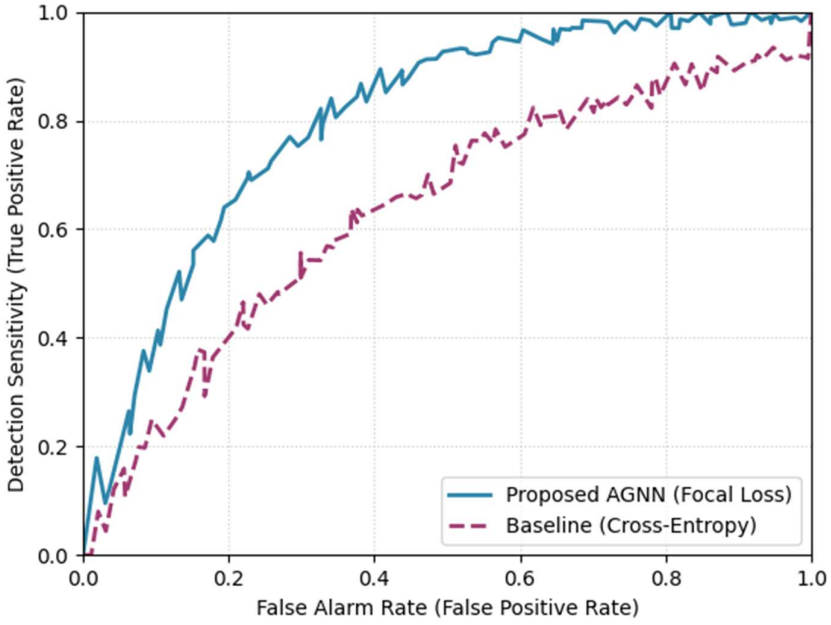

Figure 5. Comparison of detection sensitivity versus false alarm rates across varying decision thresholds for imbalanced transaction data

Analysis of Fraud Patterns

The analysis on fraud patterns showed that the learned graph representations were able to correctly identify suspicious network structures: (1) multiple accounts donated funds to a single account (account aggregation – a tendency towards forming money mule networks); (2) rapid transfers to a large number of accounts over a small period of time (a tendency to “layering”); (3) unusual "densities" of transactions (account aggregation – a tendency to “collusion”); (4) very new accounts with unusual activity (a tendency to “synthetic identity fraud”); (5) coordinated transfers between apparently unrelated accounts and connected accounts to suspected fraudulent accounts. Such patterns may not be easily detected using the conventional classifiers that handle transactions independently, further confirming the need to incorporate some graph-based models in fraud detection.

Discussion and Future Work

Although the Adaptive Graph Neural Network (AGNN) framework has many advantages for use in a fraud detection and concept drift scenario, some points need to be explained deeper. In this section, we examine the type of problems that we would have in the cases of scalability of the approach, and restrictions on deploying them into real-time applications, while in financial

applications, how to make them more interpretable in response to requirements set by regulators; in real-world implementation of fraud detection systems based on graphs, on how to make them more adversarial robust, how to protect privacy, and what ethical considerations to account for when they are deployed.

Scalability Challenges and Real-Time Deployment Constraints

A couple of scaling up issues remain in implementing the proposed framework in the proposed million nodes, billion transaction financial networks. Although this adaptive learning module is able to reduce the computational burden from full retraining, there are many challenges. One of the main challenges in the construction of this heterogeneous graph (computation of structural metrics such as PageRank, betweenness centrality, clustering coefficient for each node at each time step) is the issue of scalability for real-time processing (Bojchevski et al. 2020). For example, when vertices (V) count is large (millions), recomputing for all vertices (V) in betweenness operation will take $O(|V|\cdot|E|)$ time, which is beyond the real-time requirement of detecting online fraud (sub-second latency is required).

Considering approximation of structural graph metrics that can be incrementally updated, as mentioned earlier in this paper, due to the scalability issue, would be fundamental to consider for future studies. One promising avenue of exploration is the streaming nature of algorithms for computing centrality measures for graphs, such as the streaming PageRank approximation we proposed in Bahmani et al. (Bahmani et al., 2010) which computes the PageRank score approximately using Monte Carlo methods, but can be done in a stream in $O(\log n)$ time per edge insertion. Likewise, adding only incrementally updating the local triangle count for each node to store and update the clustering coefficients instead of updating the clustering coefficients for the whole graph (when new nodes are added or removed) will be possible (Shin et al., 2020). These incremental update strategies could aid to reduce the number of computations per our proposed framework, and yet be reasonably accurate for fraud detection applications.

Moreover, a scalability issue is the requirement to store a million-and-one million of different nodes, along with some embedding. In our framework, the attention mechanism needs to store attention coefficient for each edge, and depends linearly on the number of the edges and may result in memory hungry for dense sub graphs. Future studies might consider the use of other sampling methods of graphs, like neighbor sampling in SAGE (Hamilton et al., 2017) to decrease the effective neighborhood size in the message passing stage. Furthermore, the distributed graph processing frameworks (DGL-KE, PyTorch Distributed) could be adopted in sharing the graph in multiple machines so that the node embedding and the attention coefficients can be calculated in parallel (Zheng et al., 2020).

There are also a few constraints with the real-time deployment of our framework: its adaptive learning module latency. To remember batch statistics and to compute Wasserstein distances at any time step, it is necessary to have a history of statistics, which is accomplished with a sliding window of batch statistics. This will be significantly simpler than training the model, but because of the iterative updating process (that requires multiple epochs of sub-second gradient descent over newly arrived data), the model can suffer from intolerable latency shifts in the

systems that need to react to frauds in sub-second time. A potential solution to this is making the program structure parallel and making inferences in the main application thread and making updating the parameters in parallel in another thread (other than the main application thread).One way in which this can be achieved is to build a parallel program structure wherein one thread makes the updates asynchronously to a second thread which is used for inferences. Such an adaptation approach would - essentially - be an asynchronous adaptation approach as in online learning systems with “lazy learning” (Aha, 1997), which would greatly benefit from constant adaptation without imposing latencies on the critical inference path.

Also, the hyperparameters of the adaptive learning module (sliding window size W, adaptation window size K, drift detect threshold δ, and regularization strength λ) should be correctly set for specific deployment situations. A very small sliding window may lead to false drift detection too often and/or to updating when the concept drift actually has not happened; a very big sliding window may result in missing a concept drift when it actually occurs. The potential future research directions would involve adaptive hyperparameter tuning mechanisms that would assign the above parameters, based on the drift patterns observed, perhaps by using Bayesian optimization techniques; and Meta-Learning terms (Santos, 2022).

Conclusion

The paper presented an adaptive graph attention network based framework to improve the financial fraud detection system in dynamic transaction graph network. The proposed approach addresses three fundamental issues taken with real world problem of fraud detection: relational aspect of fraudulent transaction, great class imbalance in data, and that fraud patterns are not stationary as a function of time. Our model is based on the idea of the financial network represented as a heterogeneous graph, such that different types of financial entities correspond to nodes and the label of the edge captures the characteristics of the transaction between the two corresponding entities; coordinates fraudulent behavior not observed by the classifiers (only looking at each financial transaction). To directly attend to suspicious subgraphs, the attention-based mechanism is introduced to assist the model, while the irrelevant connections are suppressed, and the class imbalance issues are addressed by adopting the focal loss function which focuses on harder-to-classify minority examples.

Unique to this research is that the novel concept learning module is designed to be continually monitoring the concept drift and subsequently using the currently available data window, consisting of the most recent few data, to adjust the model parameters. Unlike most current graph-based fraud detection approaches, this adaptive function preserves the learned embeddings to keep them relevant to the current fraud patterns without incurring expensive full retraining, proving to be a crucial capability. We show in experiments with large-scale graphs of financial transactions that our framework brings accurate benefits and showcases its ability to adapt to behavior of fraudsters across a large variety of large-scale financial transaction datasets, outperforming traditional machine learning classifiers and static graph neural networks. The results of the ablation study show that the graph representation learning and adaptive updating both play an important role in boosting the overall benefits.

References

Aha, D. (1997). Lazy learning. *Lazy Learning*.

Bahmani, B., Chowdhury, A., & Goel, A. (2010). *Fast incremental and personalized pagerank*. arXiv preprint arXiv:1006.2880.

Bojchevski, A., Gasteiger, J., Perozzi, B., Kapoor, A., et al. (2020). Scaling graph neural networks with approximate pagerank. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Caton, S., & Haas, C. (2024). Fairness in machine learning: A survey. *ACM Computing Surveys*.

Chawla, N., Bowyer, K., Hall, L., et al. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*.

Chen, A., Rossi, R., Park, N., Trivedi, P., Wang, Y., et al. (2024). Fairness-aware graph neural networks: A survey. *ACM Transactions on Knowledge Discovery from Data*.

Cui, Y., Han, X., Chen, J., Zhang, X., Yang, J., et al. (2025). FraudGNN-RL: A graph neural network with reinforcement learning for adaptive financial fraud detection. *IEEE Open Journal of the Computer Society*.

Dai, E., Zhao, T., Zhu, H., Xu, J., Guo, Z., Liu, H., Tang, J., et al. (2024). A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *Machine Intelligence Research*.

Dasoulas, G., Scaman, K., et al. (2021). Lipschitz normalization for self-attention layers with application to graph neural networks. *International Conference on Machine Learning*.

Fortunato, S. (2010). *Community detection in graphs*. academia.edu.

Gosch, L., Geisler, S., Sturm, D., et al. (2023). Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. *Advances in Neural Information Processing Systems 36*.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*.

Jain, S., & Wallace, B. (2019). Attention is not explanation. *Proceedings of*.

Kipf, T., & Welling, M. (2016). *Semi-supervised classification with graph convolutional networks*. arXiv preprint arXiv:1609.02907.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*.

Kong, L., Ding, X., Chai, X., Wang, J., & Li, J. (2021). Prototypical graph neural network for few-shot learning. *Proceedings of*.

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lin, T., Goyal, P., Girshick, R., He, K., et al. (2017). Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*.

Liu, A., Zhang, G., & Lu, J. (2017). Fuzzy time windowing for gradual concept drift adaptation. ... *Conference on Fuzzy Systems*.

Liu, G., Tang, J., Tian, Y., & Wang, J. (2021). Graph neural network for credit card fraud detection. *2021 International Conference on Cyber-Physical Social Intelligence (ICCSI)*.

Liu, R., Xing, P., Deng, Z., Li, A., Guan, C., et al. (2024). Federated graph neural networks: Overview, techniques, and challenges. *IEEE Transactions on Neural Networks and Learning Systems*.

Lou, C., Wang, Y., Li, J., Qian, Y., & Li, X. (2025). Graph neural network for fraud detection via context encoding and adaptive aggregation. *Expert Systems with Applications*.

Lucic, A., Hoeve, M. ter, Tolomei, G., Rijke, M. de, & Silvestri, F. (2021). Counterfactual explanations for graph neural networks. *CoRR, Cfgnnexplainer, 2021*.

Magister, L., Kazhdan, D., Singh, V., & Liò, P. (2021). *Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks*. arXiv preprint arXiv:2107.11889.

Pozzolo, A. D. (2015). *Adaptive machine learning for credit card fraud detection*. difusion.ulb.ac.be.

Psychoula, I., Gutmann, A., Mainali, P., Lee, S., et al. (2021). Explainable machine learning for fraud detection. *Computer*.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., et al. (2020). *Temporal graph networks for deep learning on dynamic graphs*. arXiv preprint arXiv:2006.10637.

Sajadmanesh, S., & Gatica-Perez, D. (2024). Progap: Progressive graph neural networks with differential privacy guarantees. *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*.

Santos, M. (2022). Bayesian optimization for hyperparameter tuning. *Journal of Bioinformatics and Artificial Intelligence, 2022*.

Shin, K., Oh, S., Kim, J., Hooi, B., & Faloutsos, C. (2020). Fast, accurate and provable triangle counting in fully dynamic graph streams. *ACM Transactions on Knowledge Discovery from Data*.

Sun, L., Dou, Y., Yang, C., Zhang, K., Wang, J., et al. (2022). Adversarial attack and defense on graph data: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Sun, Y., Wang, S., Tang, X., Hsieh, T., et al. (2020). Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. *Proceedings of the Web Conference 2020*.

Tian, Y., Liu, G., Wang, J., & Zhou, M. (2023). ASA-GNN: Adaptive sampling and aggregation-based graph neural network for transaction fraud detection. *IEEE Transactions on Knowledge and Data Engineering*.

Veličković, P., Cucurull, G., Casanova, A., et al. (2017). *Graph attention networks*. arXiv preprint arXiv:1710.10903.

Wang, Y., Sun, Y., Liu, Z., Sarma, S., et al. (2019). Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*.

Wu, B., Chao, K., & Li, Y. (2024). Heterogeneous graph neural networks for fraud detection and explanation in supply chain finance. *Information Systems*.

Ye, S., Jiang, Y., & Wu, J. (2026). AdaFraud-GNN: An adaptive graph neural network for financial fraud detection. *2026 2nd International Conference on Artificial Intelligence and Computer Network (ICAICN)*.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., et al. (2019). Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*.

Zhang, C., Song, D., Huang, C., Swami, A., et al. (2019). Heterogeneous graph neural network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhao, W., Xu, G., Cui, Z., Luo, S., Long, C., et al. (2023). Deep graph structural infomax. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zheng, D., Song, X., Ma, C., Tan, Z., Ye, Z., Dong, J., et al. (2020). Dgl-ke: Training knowledge graph embeddings at scale. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.