GRANDE : a neural model over directed multigraphs with application to anti-money laundering

Ruofan Wu*
Ant Group
ruofan.wrf@antgroup.com

Boqun Ma*
Ant Group
boqun.mbq@antgroup.com

Hong Jin
Ant Group
jinhong.jh@antgroup.com

Wenlong Zhao
Ant Group
chicheng.zwl@antgroup.com

Weiqiang Wang
Ant Group
weiqiang.wwq@antgroup.com

Tianyi Zhang
Ant Group
zty113091@antgroup.com

***Abstract*—The application of graph representation learning techniques to the area of financial risk management (FRM) has attracted significant attention recently. However, directly modeling transaction networks using graph neural models remains challenging: Firstly, transaction networks are *directed multigraphs* by nature, which could not be properly handled with most of the current off-the-shelf graph neural networks (GNN). Secondly, a crucial problem in FRM scenarios like anti-money laundering (AML) is to identify risky transactions and is most naturally cast into an *edge classification* problem with *rich* edge-level features, which are not fully exploited by the prevailing GNN design that follows node-centric message passing protocols. In this paper, we present a systematic investigation of design aspects of neural models over directed multigraphs and develop a novel GNN protocol that overcomes the above challenges via efficiently incorporating directional information, as well as proposing an enhancement that targets edge-related tasks using a novel message passing scheme over an extension of edge-to-node dual graph. A concrete GNN architecture called GRANDE is derived using the proposed protocol, with several further improvements and generalizations to temporal dynamic graphs. We apply the GRANDE model to both a real-world anti-money laundering task and public datasets. Experimental evaluations show the superiority of the proposed GRANDE architecture over recent state-of-the-art models on dynamic graph modeling and directed graph modeling.**

I. Introduction

Recent years have witnessed an increasing trend of adopting modern machine learning paradigms to financial risk management (FRM) scenarios [25]. As a typical use case in operational risk scenarios like fraud detection and anti-money laundering, the identification of risky entities (user accounts or transactions) is cast into a supervised classification problem using behavioral data collected from the operating financial platform [6], [20]. For institutions like commercial banks and online payment platforms, the most important source of behavior information is the *transaction records* between users. making *transaction networks* (with users as nodes and transactions as edges) a direct and appropriate data model. Unlike standard pattern recognition tasks like image recognition where decisions are made according to information of individual objects, identification of risky patterns over transaction network requires reasoning beyond any individual scope. The phenomenon is particularly evident in the area of anti-money laundering (AML), where suspicious transactions are usually related by several users or accounts, with transactions between them being highly correlated, thereby exhibiting a cascading pattern which makes i.i.d. approaches in machine learning unsuitable. The surging developments of machine learning models over graphs, especially graph representation learning [11], have attracted significant attention in the financial industry and have shown promising results in the area of FRM [23], [21]. The dominant practice in graph representation learning is to utilize the panoply of graph neural networks (GNN) [3] that produce node-level representations via principled aggregation mechanisms which are generally described via message passing protocols [9] or spectral mechanisms [16].

Despite their convincing performance, the majority of the existent GNN models operate over *undirected graphs*, which makes them inadequate for the direct modeling of transaction networks. Firstly, many graphs that arise in FRM applications are directed by nature: i.e., in the case of a transaction network with users as nodes and transactions as edges, the direction of an edge is typically understood as the direction of its corresponding cash flow. In areas like anti-money laundering (AML), directional information is generally perceived to be of significant importance and shall not be neglected [28]. Secondly, there might exist multiple transactions between

* Equal contribution

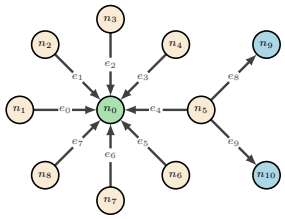

Fig. 1: Illustration on the deficiency of the directed message passing protocol in [3]: suppose the node of interest is n_0, using GNNs designed according to the protocol in [3], it becomes impossible for n_0 to aggregate information of n_9 and n_{10}. Under the context of financial risk management, suppose n_9 and n_{10} corresponds to known fraudsters, and edges correspond to transactions. Although the riskiness of n_5 might be undetermined, the transaction pattern makes it highly suspicious and therefore uplifts the riskiness of n_0. To build models that behave coherently with the above reasoning process, GNN protocols that aggregates information from *both directions* are required

certain pairs of users. Thirdly, transactions are naturally associated with timestamps that indicate the time of occurrence. Therefore, to fully utilize the graphical structure of transaction networks, we need representation learning frameworks that support *temporal directed multigraphs*. While recent progress on *dynamic graph neural networks* [38], [15] provide appropriate methods to handle temporality, discussions over neural architectures that supports directed multigraphs remains nascent [9], [24], [31], [30], [44].

From a practical point of view, the targeted risky entities may be either node (i.e., malicious users) or edges (i.e., suspicious transactions). Conventional GNN architectures produce node-level representations via encoding information of each node's rooted subtrees [39], making them a good fit for *user or account level* risk identifications. When the underlying task is to detect risky transactions, the prevailing practice is to present edges using a combination of node representations corresponding to both ends of edges. While such design may be adequate for tasks like link prediction, it lacks a way to effectively integrate edge-level information into the edge representation. Since financial networks usually contain rich edge-level features (i.e., detailed transaction-related information), refinements on edge-level representations are needed. For example, to accurately represent a transaction, we need to combine the information of its buyer (cash sender) and seller (cash receiver), and the transaction-related information, with each of them requiring aggregating relevant information from related users and transactions. A recent line of work [13], [4], [14] focused directly on *learned edge representations* using the idea of *edge-to-node duality* and obtained satisfactory performance over downstream tasks like edge classification. However, previous works on edge representation learning all applies to undirected graphs, making the extension to transaction networks highly non-trivial.

In this paper, we propose a general message passing neural network protocol that simultaneously outputs node and edge representations over directed multigraphs. Based on this protocol, we derive a GNN architecture called GRANDE with an extension to temporal graphs that efficiently leverages the underlying structural property of transaction networks. More specifically, we summarize our contribution as follows:

- We develop a novel bi-directional message passing protocol with duality enhancement (BiMPNN-DE) that strengthens previous proposals over message passing neural architectures over directed multigraphs. The improvement is two-fold: Firstly, it effectively combines neighborhood information from both incoming and outgoing edges of nodes. Secondly, it simultaneously outputs node and edge representations via performing message passing over both the original graph and its *augmented* edge adjacency graph.
- We derive a concrete GNN architecture following the proposed BiMPNN-DE protocol called GRANDE, that devices the acclaimed transformer [32] mechanism for neighborhood aggregation. The proposed GRANDE framework is made compatible with temporal directed multigraphs through the integration of a generic time encoding module that further extends previous works on dynamic graph modeling [38].
- To show the practical effectiveness of GRANDE, we apply it to a suspicious transaction identification task in anti-money laundering, with the underlying transaction network data collected from one of the world's leading online payment platforms. Comparisons against various undirected and directed GNN baselines show the superiority of the proposed model. We also provide evaluations on two public datasets generated from transaction networks to further verify the strength of GRANDE framework when underlying graph features are relatively weak.

II. Methodology

A. Problem formulation

Under the context of financial risk management, we consider the following *event stream* representation of recorded transaction data that are available in most online transaction systems:

$$\mathcal{E} = \{(u_1, v_1, t_1, \chi_1), (u_2, v_2, t_2, \chi_2), \ldots\} \tag{1}$$

Each event (u, v, t, χ) is interpreted as a transaction from user u to user v that occurred at time t, with related features χ that could often be further decomposed as user-level features like user account information, and event-level features like transaction amount and channels. In this paper, we focus on the representative task of *transaction property prediction* that typically takes the form of binary classification that aims at identifying illicit or fraudulent transactions. The task could be cast into a graph learning problem of edge classification in a straightforward manner. We consider the temporal graph modeling paradigm [15] that views the underlying temporal graph as being generated from the event stream $\mathcal{E}$. Therefore, given a time period $\mathcal{T} = [\tau_{\mathsf{start}}, \tau_{\mathsf{end}}]$, we construct the graph data as the snapshot $G(\mathcal{T}) = (V(\mathcal{T}), E(\mathcal{T}))$ of the underlying temporal graph. Since there may exist multiple transactions between the same set of users, we consider $G(\mathcal{T})$ to be a

directed multigraph with each edge in the edge multiset $E(\mathcal{T})$ represents an event that happens inside the time interval $\mathcal{T}$, and the node set $V(\mathcal{T})$ consists of related users corresponding to the included events. During the training stage, we construct a snapshot $G(\mathcal{T}_{\text{train}})$, and obtain a possibly incomplete set of edge labels that are understood as edge properties annotated using expert knowledge. During testing stage, we perform inference over snapshots $G(\mathcal{T}_{\text{test}})$ that are based on later time intervals than $\mathcal{T}_{\text{train}}$. We assume $\mathcal{T}_{\text{train}} \cap \mathcal{T}_{\text{test}} = \emptyset$, hence the problem of interest could be viewed as *inductive edge classification over temporal graphs*.

B. Message passing protocols and directed graphs

Let $G = (V, E)$ be a directed multigraph with node set V and edge multiset E. For any pairs of nodes (u, v), denote $\mu(u, v)$ as the number of edges going from u to v. Then G becomes a (simple) graph when $\max_{u\in V, v\in V} \mu(u, v) \leq 1$. For each $v \in V$, denote $N^+(v) = \{u, (v, u) \in E\}$ and $N^-(v) = \{u, (u, v) \in E\}$ as its out-neighborhood and in-neighborhood respectively, and let $N(v) = N^+(v) \cup N^-(v)$ be its neighborhood. For the sake of presentation clarity, we will overload the notation uv for both an edge in the undirected graph, or a directed edge from u to v in a directed (multi)graph from time to time, with its exact meaning being clear from the context. We are interested in the general case where both node features $X = \{x_v\}_{v\in V}$ and edge features $Z = \{z_{uv}\}_{(u,v)\in E}$ are available, where we assume both kinds of features to be of dimension d. In this paper we focus on neural approaches to such directed multigraphs. A good starting point is the neural message passing scheme for undirected graphs [9]: let $h_v^{(l)}$ denote the hidden representation of node v at the l-th layer of the network, and $h_v^{(0)} = x_v, \forall v \in V$. The message passing graph neural network protocol (abbreviated as GNN hereafter) is described recursively as:

$$h_v^{(l+1)} = \mathsf{COMBINE}\left(h_v^{(l)}, \mathsf{AGG}\left(\mathsf{MESSAGE}(h_v^{(l)}, h_u^{(l)}, z_{uv}, u \in N(v))\right)\right) \quad (2)$$

Different combinations of COMBINE, AGG and MESSAGE mechanisms thus form the *design space* of undirected GNNs [41]. To the best of our knowledge, there are three types of generalization strategies to directed graphs:

Symmetrization The most ad-hoc solution is to "make it undirected" via padding necessary reverse edges so that $N(v) = N^+(v) = N^-(v)$, and apply standard graph neural networks that operate on undirected graphs like GCN or GAT. Despite its simplicity and clearness, the symmetrization approach discards directional information in the digraph and may raise subtleties when dealing with multigraphs.

DiGraph-theoretic motivations A more recent line of work [24], [31], [30], [44] drew insights from directed graph theory, especially the spectral branch [7]. The proposed models are mostly digraph analogs of GCN, without the consideration for edge features, therefore severely limiting the design space of directed message passing GNNs.

Directed Protocol In the seminal work [3, Algorithm 1], the authors proposed a GNN protocol that operates on directed multigraphs with edge features via aggregating messages from only the in-neighborhood, i.e., replacing $N(v)$ in (2) with $N^-(v)$). [1] While being a natural extension, such kind of GNN protocol losses information from the outgoing direction of each node. We present a pictorial illustration in figure 1.

To address the aforementioned shortcomings, we propose a novel GNN protocol that operates on directed digraphs termed *bi-directional message passing neural network (BiMPNN)*. The protocol extends the standard undirected protocol (2) via enabling each node to aggregate information from both its in-neighborhood and out-neighborhood:

$$\begin{aligned} h_v^{(l+1)} &= \mathsf{MERGE}\left(\phi_v^{(l+1)}, \psi_v^{(l+1)}\right) \\ \phi_v^{(l+1)} &= \mathsf{COMBINE}_{\mathsf{in}}\Big(h_v^{(l)}, \\ &\qquad \mathsf{AGG}_{\mathsf{in}}\left(\mathsf{MESSAGE}_{\mathsf{in}}(h_v^{(l)}, h_u^{(l)}, z_{uv}, u \in N^-(v))\right)\Big) \\ \psi_v^{(l+1)} &= \mathsf{COMBINE}_{\mathsf{out}}\Big(h_v^{(l)}, \\ &\qquad \mathsf{AGG}_{\mathsf{out}}\left(\mathsf{MESSAGE}_{\mathsf{out}}(h_v^{(l)}, h_r^{(l)}, z_{vr}, r \in N^+(v))\right)\Big) \end{aligned} \quad (3)$$

Despite being more complicated than the undirected protocol, the proposed BiMPNN protocol remains conceptually clear: for each layer, we aggregate separately from each node's in-neighborhood and out-neighborhood using distinct aggregation mechanisms, and merge the two obtained intermediate representations into the next layer's input. Consequently, a k-layer GNN derived from the BiMPNN protocol utilizes both its *root-k incoming subtree* and outgoing subtree, thereby providing a richer set of *relational inductive bias* than the one proposed in [3].

C. Edge-level task and edge-to-node duality

The BiMPNN protocol (3) provides a principled way of obtaining node representations in directed multigraphs which serves as the building block of *node-level tasks*. Yet another important type of graph-related tasks (in a *local* sense [27]) is *edge-level* tasks, which exhibits a dichotomy between *edge-existence* prediction, i.e., link prediction, and *edge-property* prediction, i.e., edge classification. For the later task type, the very existence of an edge itself suggests basing the predictions on a properly defined *edge representation*, which should go beyond naively concatenating node representations of its ends [14]. Although the protocol (3) implicitly encodes edge features into node representations, it ignores the cascading dynamics of edges (i.e., information implied by cash flow in FRM applications). To build powerful edge representations that efficiently adapt to the underlying graph structure. Mechanisms based on the edge-to-node dual graphs, or *line graph*s, have been proposed [5], [13], [4], [14] under the undirected GNN protocol (2). To begin our discussion on possible extensions to directed multigraphs, we first review the definition of line graphs as follows:

[1] The original version also considered incorporation of a *global node* that aggregates information from the whole graph regardless of the connectivity structure. While such design choice may have some gains in moderate size graphs [36], it does not scale to large graphs. Therefore we will not consider such design choice in this paper

Definition 1 (Line graph and Line digraph [10], [2])**.** For both undirected graph and directed (multi)graphs where we overload notation without misunderstandings, $G = (V, E)$, the node set of its line graph $L(G) = (L(V), L(E))$ is defined as its edge (multi)set $L(V) = E$.

Undirected graph the edge set of its line graph is defined as

$$L(E) = \{(uv, rs) : (u, v) \in E, (r, s) \in E, \{u, v\} \cap \{r, s\} \neq \emptyset\} \quad (4)$$

Directed (multi)graph the edge set of its line graph is defined as

$$L(E) = \{(uv, rs) : (u, v) \in E, (r, s) \in E, v = r\} \quad (5)$$

For undirected graphs, their line graphs provides a natural way to update edge representations under standard message passing protocols like (2). However, trivially extending (3) using the definition of line digraphs may incur significant information loss: We take the graph in figure 1 as an example, its line graph has an *empty* edge set, which makes the message passing framework useless over the derived line graph. While the graph-theoretic definition enjoys some nice properties [2], the adjacency criterion might be overly stringent for deriving useful GNN architectures. Intuitively, we may expect different transactions triggered by the same account as correlated rather than independent, which makes connectivity of edges like (n_5, n_9) and (n_5, n_0) as desirable. Therefore, we propose the following *augmentation strategy* to obtain an *augmented edge adjacency graph* $\overline{L(G)} = (\overline{L(V)}, \overline{L(E)}, T(E))$: The node set is still defined as $\overline{L(V)} = E$, and we augment the edge set using the undirected adjacency criterion (4). To retain directional information, we encode the adjacency pattern of two edges (with four possible patterns: *head-to-head*, *head-to-tail*, *tail-to-head*, *tail-to-tail*) into a categorical vector, which we denote as $T(E) = \{\mathsf{type}(uv, rs) : (uv, rs) \in \overline{L(E)}\}$. By construction, for each edge in $\overline{L(E)}$, its reverse is also in $\overline{L(E)}$ with possibly different edge types. We provide a pictorial illustration in the left part of figure 2.

To derive an edge representation update rule, we follow the spirit of the BiMPNN node update rule (3): let $N_L^+(uv), N_L^-(uv)$ be the out and in neighborhoods in the ordinary line graph $L(G)$ of G, and $\overline{N_L^+(uv)}, \overline{N_L^-(uv)}$ to be those in $\overline{L(G)}$, respectively. For each edge $(uv, rs) \in \overline{L(E)}$, we use $\mathsf{C}(uv, rs) \in V$ as the common incident node of the edges uv and rs. The following updating rule enhances BiMPNN protocol with duality information, which we term BiMPNN-DE:

$$\begin{aligned}
h_v^{(l+1)} &= \mathsf{MERGE}^{\mathsf{node}}\left(\phi_v^{(l+1)}, \psi_v^{(l+1)}\right) \\
g_{uv}^{(l+1)} &= \mathsf{MERGE}^{\mathsf{edge}}\left(\theta_{uv}^{(l+1)}, \gamma_{uv}^{(l+1)}\right) \\
\phi_v^{(l+1)} &= \mathsf{COMBINE}_{\mathsf{in}}^{\mathsf{node}}\Big(h_v^{(l)}, \\
&\qquad \mathsf{AGG}_{\mathsf{in}}^{\mathsf{node}}\left(\mathsf{MESSAGE}_{\mathsf{in}}^{\mathsf{node}}(h_v^{(l)}, h_u^{(l)}, g_{uv}^{(l)}, u \in N^-(v))\right)\Big) \\
\psi_v^{(l+1)} &= \mathsf{COMBINE}_{\mathsf{out}}^{\mathsf{node}}\Big(h_v^{(l)}, \\
&\qquad \mathsf{AGG}_{\mathsf{out}}^{\mathsf{node}}\left(\mathsf{MESSAGE}_{\mathsf{out}}^{\mathsf{node}}(h_v^{(l)}, h_r^{(l)}, g_{vr}^{(l)}, r \in N^+(v))\right)\Big) \\
\theta_{uv}^{(l+1)} &= \mathsf{COMBINE}_{\mathsf{in}}^{\mathsf{edge}}\Big(g_{uv}^{(l)}, \\
&\qquad \mathsf{AGG}_{\mathsf{in}}^{\mathsf{edge}}\left(\mathsf{MESSAGE}_{\mathsf{in}}^{\mathsf{edge}}(g_{uv}^{(l)}, g_{pq}^{(l)}, \hat{h}_{pq,uv}^{(l)}, pq \in \overline{N_L^-(uv)})\right)\Big) \\
\gamma_{uv}^{(l+1)} &= \mathsf{COMBINE}_{\mathsf{out}}^{\mathsf{edge}}\Big(g_{uv}^{(l)}, \\
&\qquad \mathsf{AGG}_{\mathsf{out}}^{\mathsf{edge}}\left(\mathsf{MESSAGE}_{\mathsf{out}}^{\mathsf{edge}}(g_{uv}^{(l)}, g_{rs}^{(l)}, \hat{h}_{uv,rs}^{(l)}, rs \in \overline{N_L^+(uv)})\right)\Big) \\
\hat{h}_{uv,rs}^{(l)} &= \mathsf{COMBINE}^{\mathsf{type}}\left(h_{\mathsf{C}(uv,rs)}^{(l)}, T_{\mathsf{type}(uv,rs)}\right)
\end{aligned} \quad (6)$$

Where we use $g_{uv}^{(l)}$ to denote the hidden representation of edge uv at the l-th layer of GNNs derived from the BiMPNN-DE protocol. The protocol devices an additional edge representation update component that mirrors the BiMPNN protocol over the augmented edge adjacency graph $\overline{L(G)}$ (see the last four equations in the display (6)). To obtain an edge representation counterpart $\tilde{h}_{uv,rs}^{(l)}$ during the aggregation process over $\overline{L(G)}$, we use an additional $\mathsf{COMBINE}^{\mathsf{type}}$ mechanism that combines features of the common incident node and the information of adjacent types, with is encoded into a learnable edge type embedding matrix $T \in \mathbb{R}^{4\times d}$. The BiMPNN-DE protocol (6) offers a much larger design space than that of BiMPNN protocol. In its full generality, we may specify up to 15 different mechanisms corresponding to different MERGE, COMBINE, AGG and MESSAGE operations. From a practical point of view, we may design the aforementioned operations using parameterized functions that share the same underlying structure.

D. The GRANDE architecture

In this section, we devise the previously developed BiMPNN-DE protocol (6) to derive a concrete GNN architecture that simultaneously outputs node and edge representations, along with an improvement strategy that targets edge-property prediction tasks. We base our design upon the acclaimed Transformer architecture [32], which has seen abundant adaptations to GNNs recently [38], [8], [40]. We define the multiplicative attention mechanism that incorporates edge information as follows:

$$\begin{aligned}
\mathsf{ATTN}(h_v, \{h_u, g_{uv}\}_{u\in N(v)}) &= \sum_{u\in N(v)\cup\{v\}} \alpha_{uv} W_N h_u + \beta_{uv} W_E g_{uv} \\
\alpha_{uv} &= \frac{\exp\left(\langle W_Q h_v, W_K h_u\rangle/\sqrt{d}\right)}{\sum_{u\in N(v)\cup\{v\}} \exp\left(\langle W_Q h_v, W_K h_u\rangle/\sqrt{d}\right)} \\
\beta_{uv} &= \frac{\exp\left(\langle W_Q h_v, W_E g_{uv}\rangle/\sqrt{d}\right)}{\sum_{u\in N(v)\cup\{v\}} \exp\left(\langle W_Q h_v, W_E g_{uv}\rangle/\sqrt{d}\right)}
\end{aligned} \quad (7)$$

We include commonly used operations in a transformer block, namely LayerNorm (LN), skip connection and a learnable two

layer MLP (FF) as nonlinearity [32], and wraps them into a transformer block:

$$\begin{aligned}\textsf{TRANSFORMER}(h_v, \{h_u, g_{uv}\}_{u\in N(v)}) &= \textsf{LN}(\tilde{h}_v + \textsf{FF}(\tilde{h}_v)) \\ \tilde{h}_v &= \textsf{LN}(h_v + \textsf{ATTN}(h_v, \{h_u, g_{uv}\}_{u\in N(v)}))\end{aligned} \tag{8}$$

After defining the basic mechanisms, we write the node and edge update rules as follows:

$$\begin{aligned}
h_v^{(l+1)} &= \textsf{CONCAT}\left(\phi_v^{(l+1)}, \psi_v^{(l+1)}\right) \\
g_{uv}^{(l+1)} &= \textsf{CONCAT}\left(\theta_{uv}^{(l+1)}, \gamma_{uv}^{(l+1)}\right) \\
\phi_v^{(l+1)} &= \textsf{TRANSFORMER}_{\textsf{in}}^{\textsf{node}}\left(\Phi_N h_v^{(l)}, \{\Phi_N h_u^{(l)}, \Phi_E g_{uv}^{(l)}\}_{u\in N^-(v)}\right) \\
\psi_v^{(l+1)} &= \textsf{TRANSFORMER}_{\textsf{out}}^{\textsf{node}}\left(\Psi_N h_v^{(l)}, \{\Psi_N h_r^{(l)}, \Psi_E g_{vr}^{(l)}\}_{r\in N^z=(v)}\right) \\
\theta_{uv}^{(l+1)} &= \textsf{TRANSFORMER}_{\textsf{in}}^{\textsf{edge}}\left(\Theta_N g_{uv}^{(l)}, \{\Theta_N g_{uv}^{(l)}, \Theta_E \hat{h}_{pq,uv}^{(l)}\}_{pq\in\overline{N_L^-(uv)}}\right) \\
\gamma_{uv}^{(l+1)} &= \textsf{TRANSFORMER}_{\textsf{out}}^{\textsf{edge}}\left(\Gamma_N g_{uv}^{(l)}, \{\Gamma_N g_{uv}^{(l)}, \Gamma_E \hat{h}_{uv,rs}^{(l)}\}_{rs\in\overline{N_L^+(uv)}}\right)
\end{aligned} \tag{9}$$

The updating equations (9) involve many learnable parameters, to which we apply the following naming convention: We use upper case greek letters $\Phi, \Psi, \Theta, \Gamma$ to denote projection matrices that takes value in $\mathbb{R}^{2d\times d}$, and we use the subscript N for *node-related projection* and E for *edge-related projection*. The $\textsf{COMBINE}^{\textsf{type}}$ operation is set to be element-wise addition.

Improvements over edge-property prediction tasks The architecture in (9) is most helpful when the underlying task is to predict properties of some existent edges, which serves as the underlying task for many financial applications like fraud detection and AML. Toward this goal, an ad-hoc solution is to base the prediction with respect to edge (u, v) upon the concatenation of h_u, h_v and g_{uv}. According to recent practices over pairwise learning on graphs [34], [35], incorporation of interactions between $N(u)$ and $N(v)$, or *cross node interactions* significantly improves prediction performance. However, the cross-node attention module proposed in [34] requires full attention between $h_{N(u)}$ and $h_{N(v)}$, yielding a potentially large computation overhead. Here we provide a more efficient alternative to model the cross-node interactions called the *cross-query attention* module. Given a node pair (u, v), compute

$$\begin{aligned}
\delta_{uv} &= \textsf{CONCAT}\Big(\textsf{ATTN}_{\textsf{left}}^{\textsf{in}}(h_u, \{h_r, g_{rv}\}_{r\in N^-(v)}), \\
&\qquad\qquad \textsf{ATTN}_{\textsf{left}}^{\textsf{out}}(h_u, \{h_s, g_{vs}\}_{s\in N^+(v)})\Big) \\
\delta_{vu} &= \textsf{CONCAT}\Big(\textsf{ATTN}_{\textsf{right}}^{\textsf{in}}(h_v, \{h_r, g_{ru}\}_{r\in N^-(u)}), \\
&\qquad\qquad \textsf{ATTN}_{\textsf{right}}^{\textsf{out}}(h_v, \{h_s, g_{us}\}_{s\in N^+(u)})\Big)
\end{aligned} \tag{10}$$

The above procedure (10) performs four queries that attends u and v to its opponent's in and out neighbors, respectively. Providing the existence of an edge (u, v), the mechanism (10) could be understood as attending nodes to the a specific subsets of their *second-order* neighborhoods which have close relationship to the edges of interest. Finally we summarize the previous developments into a framework termed multi**G**raph t**RAN**sformer with **D**uality **E**nhancement (GRANDE) , that utilizes the following edge representation for edge-property prediction tasks:

$$g_{uv}^{\textsf{GRANDE}} = \textsf{CONCAT}(g_{uv}, h_v, h_u, \delta_{uv}, \delta_{vu}) \tag{11}$$

E. Extension to temporal graphs

According to the formulation in section II-A, extending the GRANDE framework to $G(\mathcal{T})$ requires utilization of the edge-wise timestamp information $\{t_{uv}, (u, v) \in E\}$ to produce *time-aware node and edge representations*. Inspired by recent developments of time-aware representation learning approaches [37], [38], we propose an *extended generic time encoding* mechanism that enhances the node and edge update rule of BiMPNN-DE with temporal information. Finally, to better exploit the structure of the temporal graph, we suggest a *pruning strategy* that shrinks the number of edges of $\overline{L(G_{\mathcal{T}})}$ by a factor up to two.

Generic time encoding The functional time encoding (FTE) provides a principled way that modifies the self-attention operation (7) with minimal architecture change:

$$\begin{aligned}
&\textsf{TATTN}(h_v, \{h_u, g_{uv}\}_{u\in N(v)}) \\
&\quad= \sum_{u\in N(v)\cup\{v\}} \alpha_{uv} W_N \check{h}_{uv} + \beta_{uv} W_E \check{g}_{uv} \\
&\check{h}_{uv} = \textsf{CONCAT}(h_u, \textsf{TE}(|\Delta t_{uv}|)) \\
&\check{g}_{uv} = \textsf{CONCAT}(g_{uv}, \textsf{TE}(|\Delta t_{uv}|))
\end{aligned} \tag{12}$$

Where $\textsf{TE}(|\Delta t_{uv}|)$ is temporal embedding obtained via certain FTE mechanisms. α_{uv}s and β_{uv}s are calculated analogously to the rules in (7). In this paper we will follow the Bochner-type FTE [37], [38]:

$$\textsf{TE}(s) = \sqrt{\tfrac{1}{d}}\left[\cos(\rho_1 s), \sin(\rho_1 s), \ldots, \cos(\rho_d s), \sin(\rho_d s)\right] \tag{13}$$

with learnable parameters $(\rho_1, \ldots, \rho_d)$. According to the setup in [38], timestamps are associated with nodes rather than edges, hence the authors used the time difference $\Delta t_{uv} = t_u - t_v$. The situation becomes more complicated when timestamps are associated with edges, for which we propose the following modification:

$$\Delta t_{uv} = t_{uv} - \min_{u\in N(v)} t_{uv} \tag{14}$$

We use FTE based on (14) and replace the ATTN component with TATTN component of the node time difference calculation in the node update part of (9). For the edge update part, the ordinary node time difference method applies naturally since edge timestamps act as node timestamps in the dual formulation.

Temporal information and pruning strategies the proposed line graph augmentation strategy produces an undirected $\overline{L(G)}$. While this undirected network might provide valuable information in general, in financial risk management scenarios where edges represent *timestamped transactions*, directed adjacencies between transactions are typically of interest: Consider a transaction e_{t_0} from Bob to Alice, it is reasonable to assume that *downstream* transactions of e_{t_0} (i.e., transactions e_t with $t > t_0$) will be affected by e_{t_0}, but its upstream transactions shall not be affected. Inspired from this intuition, we introduce a *causal pruning strategy* that applies when temporal information is available in the underlying graph, i.e, for an edge e that goes from u to v,

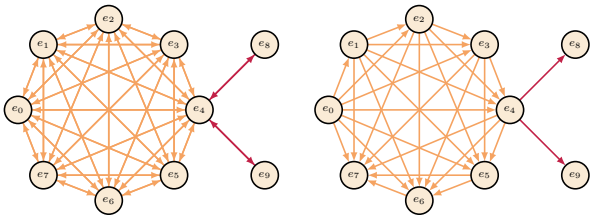

Fig. 2: An illustration of the proposed line graph augmentation strategy: The left figure stands for the augmented edge adjacency graph $\overline{L(G)}$ for the digraph depicted in figure 1. We use colored edges to represent edge types: head-to-head and tail-to-tail, note that the remaining two kinds of edge types do not appear in $\overline{L(G)}$. The right figure shows the effect of the causal pruning strategy under the additional temporal constraint that $t_0 < t_1 < \cdots < t_9$, with t_i being the occurrence time of edge t_i for $i \in \{0, \ldots, 9\}$

we have its time of occurrence t_e. The causal pruning strategy deletes edges in $\overline{L(G)}$ with the occurrence time of the head node being earlier than that of the tail node. When the causal pruning strategy is applicable, we may prune up to 50% of the edges in $\overline{L(G)}$. An illustration is provided in the right part of figure 2. Note that the proposed strategy is closely related to the construction of causal temporal subgraphs in temporal graph modeling literature [38], [29].

F. Scalability and complexity

Most of the real-world financial networks like transaction networks are *sparse*, i.e., most people only make transactions to a few others given a finite time window. Consequently, the computational complexity of any message passing neural networks could be roughly regarded as $O(Ed^2)$. Extending ordinary MPNN architectures to BiMPNN protocol doubles the computation cost, which could be easily resolved through parallelization in modern deep learning frameworks like tensorflow [1]. The extra computational cost brought by introducing the dual component (6) requires more care: even when the original graph is sparse, its augmented edge adjacency graph might be dense or even complete. Such cases do happen in realistic scenarios since large hubs frequently exist in transaction networks, which corresponds to a complete subgraph in the dual network. Therefore the worst-case computation cost of $O(E^2d^2)$ is sometimes inevitable in architectures derived from the BiMPNN-DE protocol (6). Hence to meet the computational requirement of GNN architectures like GRANDE , performing GNN training/inference over the whole graph is unrealistic. Instead, we resort to a *local* computation alternative implemented by the AGL system [43] which grabs the K-hop rooted subgraph of each target node and performed batched stochastic training and efficient parallel inference given distributed infrastructures like MapReduce [43]. In practical scenarios, it is often reasonable to set an upper bound M_{max} on the edges of any K-hop rooted subgraph and device proper sampling methods to meet the requirement. The resulting computational complexity during training is reduced to $O(BM_{\mathsf{max}}^2 d^2)$, where B denotes the batch size. Since we may control M_{max} so that the whole batch of subgraphs fits the storage requirement of high-performance hardware like GPU, the computational costs of running GRANDE becomes fully affordable for industry-scale distributed training and inference.

III. Related Works

A. Neural models over directed graphs

Directional extensions of message passing GNN protocol were mentioned in pioneer works [9], [3] without providing empirical evaluations. Recent developments toward designing GNNs for digraphs are mostly inspired by different types of graph Laplacians that are defined over digraphs. For example, [30] used the definition in [7] and [44] used the Hermitian magnetic Laplacian to decouple the aggregation process of graph connectivity and edge orientations.

B. GNNs for edge representation learning

The idea of utilizing node-to-edge duality was explored in early works like LGNN [5], where the authors drew insights from community detection literature, and use the non-backtracking walk operator [17] to define the dual graph and perform GCN-like aggregations simultaneously over both graphs. Later developments [13], [4], [14] focused on variants of LGNN with the alternative definition of the dual, such as the standard LINE graph [10].

C. Transformer architectures over graphs

The renowned GAT architecture [33] could be regarded as using the *additive* attention mechanism to form the attention layer, as opposed to the *multiplicative* attention mechanism adopted by the Transformer architecture [32]. Adaptations of the original Transformer to graph context have been assessed recently, [8] replaced the additive attention in GAT with inner product attention and use spectral embedding as a proxy for the positional embedding component in the original transformer architecture. In [40], the authors proposed to use *full-attention* transformers and use graph-theoretical attributes of nodes and edges to guide the attention procedure. While the results were shown competitive over biological benchmarks, the computational overhead is way too heavy for industrial-level graphical applications.

IV. Experiments

In this section, we report empirical evaluations of GRANDE over an industrial application as well as assessments over public datasets. We focus on the edge classification task over temporal directed multigraphs. Finally, we present a detailed ablation study to decompose the contributions of different constituents of GRANDE .

A. Datasets

We use one industrial dataset and two public datasets, with their summary statistics listed in table III in appendix A.

AML dataset This dataset is generated from transaction records collected one of the world's leading online payment systems. The business goal is to identify transactions that exhibit risky patterns as being highly suspicious of money

laundering. The underlying graph is constructed by treating users as nodes and transactions as directed edges with arbitrary multiplicity. We engineer both node and edge features under a two-stage process: We first obtain raw node features via statistical summaries of corresponding user's behavior on the platform during specific time periods, and raw edge features consist of transaction properties as well as related features of two users involved in the transaction. [2] The decision tree feature transforms [12] is then applied to both features so that after the transform, the input node and edge feature for all the assessed models are sparse categorical features with dimension 6400. For both training and testing, we collect data under a 10-day period with no overlap between the training period and the testing period. A random subset corresponding to 10% of the testing data is held out for validation.

Bitcoin datasets We use two who-trusts-whom networks of people who trade using Bitcoin on two different platforms, Bitcoin OTC and Bitcoin Alpha [19], [18]. Both networks are directed without edge multiplicities, each edge is associated with a timestamp and a trust score ranging from -10 to 10. We consider the task of binary edge classification with edge labels generated as whether the trust score is negative. Using node features represented as the concatenation of one-hot representation of in and out-degree of nodes. For both datasets, we use the chronological split that uses 70% data for training, 10% for validation, and 20% for testing

B. Baselines

We compare the proposed GRANDE framework with the following types of baselines:

Undirected approaches We consider two representative GNN architectures GCN [16] and GAT [33] that operate on undirected graphs. Since temporal information is available in all three datasets, we also include the TGAT architecture [38]. As all the aforementioned methods produce node-level representations, we use the concatenation of node representations as edge representation according to the adjacency structure. As frameworks that directly output edge representation remain few, we include the EHGNN architecture [14] as a strong baseline. To make the undirected architectures compatible with directed (multi)graphs, we add reverse edges with duplicated edge features if there exist no edge multiplicities in the digraph. Otherwise, we keep only one edge between each pair of nodes, with the corresponding edge feature generated via aggregating the original edge features (according to the "multigraph to graph" hierarchy) using the DeepSet method [42], and add reverse edges thereafter.

DiGraph-oriented approaches We consider two digraph GNN architectures that utilizes different notions of directed graph Laplacians, DGCN [24] and MagNet [44].

The aforementioned baselines exclude some of the recently proposed state-of-the-art GNN models like Graphormer [40] for undirected graphs or directed approaches like DiGCN [30] due to scalability issues, i.e., they require either full graph attention or solving eigen programs over the full graph Laplacian, which are computationally infeasible for industry-scale graphs.

C. Experimental setup

Across all the datasets and models, we use a two-layer architecture with hidden dimension $d = 128$ without further tuning. For models with generic time encodings, we fix the dimension of time encoding to be 128. For transformer related architectures, we follow the practice in [32] and use a two-layer MLP with ReLU activation with hidden dimension 512. As all the relevant tasks are binary classifications, we adopt the binary cross entropy loss as the training objective, with ℓ_2 regularization under a coefficient 0.0001 uniformly across all experiments. The graph data are constructed via the GraphFlat component of the AGL system [43] that transforms the raw graph data into batches of subgraphs with appropriate sampling. [3] We use Adam optimizer with a learning rate of 0.0001 across all tasks and models. For the bitcoin datasets, we train each model for 10 epochs using a batch size of 128 and select the best-performed one according to the roc-auc score on the validation data under periodic evaluations every 100 steps. For the AML dataset, we train the model for 2 epochs with a batch size of 256 as the size of the dataset is sufficiently large. We adopt similar model selection criterion as those of Bitcoin datasets, with periodic evalutions every 500 steps.

Metrics Since the primary focus of this paper is applications to the FRM scenario, we choose three representative metrics, namely roc-auc score (AUC), Kolmogorov-Smirnov statistic (KS) and F1 score (F1).

D. Performance

We present evaluation results in table I. Apart from the proposed GRANDE architecture, we report a *reduced* version of GRANDE via discarding all operations on the augmented edge adjacency graph, as well as the cross-query attention module (10). The resulting model could be considered as implementing a time-aware variant of graph transformer under the BiMPNN protocol. We summarize our experimental findings as follows:

- For the Bitcoin datasets which could be considered as under the *weak feature* regime, the GRANDE architecture obtains substantial performance improvement: On the Bitcoin-OTC dataset, the relative improvement over the best baselines are 10.1%, 30.7% and 22.6% with respect to AUC, KS, and F1. On the Bitcoin-Alpha dataset, the relative improvement is more significant with 19.2%, 67.3%, and 35.4% respectively. We attribute the improvements to both the directional information and the duality information that GRANDE utilizes. The improvements of the directional information

[2]Per organizational regulations, the detailed feature engineering logic is not fully described. We will consider (partially) releasing the AML dataset after passing relevant security checks of the company, as well as the source code.

[3]The AGL framework is particularly useful when dealing with industry-scale graphs that are barely possible to process as a whole. However, it may lose some information in the sampling stage of the preprocessing phase. To fully mimic the industrial setup, we preprocess all three datasets using AGL, therefore the results of Bitcoin datasets are not directly comparable to previously published results.

TABLE I: Experimental results over two public Bitcoin datasets and the AML dataset, with best performances in **bold**

	Bitcoin-OTC			Bitcoin-Alpha			AML		
	AUC	KS	F1	AUC	KS	F1	AUC	KS	F1
GCN [16]	0.742	0.376	0.432	0.626	0.198	0.282	0.958	0.793	0.704
GAT [33]	0.736	0.381	0.393	0.626	0.178	0.269	0.962	0.802	0.718
TGAT [38]	0.744	0.401	0.422	0.656	0.248	0.294	0.963	0.804	0.720
EHGNN [14]	0.719	0.356	0.420	0.626	0.228	0.297	0.961	0.804	0.718
DGCN [24]	0.634	0.243	0.354	0.633	0.228	0.282	0.962	0.806	0.720
MagNet [44]	0.753	0.388	0.434	0.645	0.217	0.293	0.954	0.780	0.688
GRANDE (reduced)	0.789	0.459	0.460	0.669	0.256	0.294	0.965	0.810	0.726
GRANDE	**0.829**	**0.524**	**0.532**	**0.769**	**0.415**	**0.398**	**0.966**	**0.813**	**0.734**

TABLE II: Performance summary of ablation studies, with best performance in **bold**

	Bitcoin-OTC			Bitcoin-Alpha			AML		
	AUC	KS	F1	AUC	KS	F1	AUC	KS	F1
GRANDE	0.829	0.524	**0.532**	**0.769**	**0.415**	**0.398**	**0.966**	**0.813**	**0.734**
- reduced	0.789	0.459	0.460	0.669	0.256	0.294	0.965	0.810	0.726
- w/o causal pruning	0.801	0.464	0.485	0.726	0.325	0.339	0.966	0.813	0.732
- w/o time encoding	**0.834**	0.525	0.531	0.691	0.265	0.324	0.966	0.813	0.731
- w/o cross-query attention	0.828	**0.528**	0.504	0.766	0.395	0.372	0.966	0.813	0.730
- w line graph	0.762	0.405	0.415	0.655	0.229	0.285	0.965	0.812	0.729

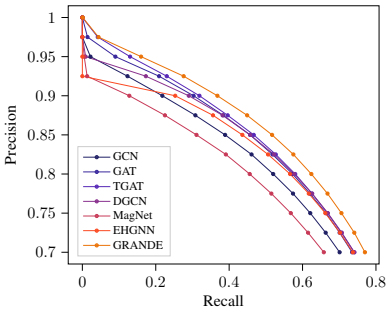

Fig. 3: Precision-recall (PR) curve under the AML dataset. We show recall values corresponding to high precision ranges (from 0.7 to 1.0 equally spaced with 0.025)

could be inferred from the results of the reduced GRANDE variant, which exhibits solid improvements over all the baselines. The incorporation of edge-to-node duality and cross-query attention systematically encodes more structural information, therefore yielding further improvements.

- For the AML dataset which could be regarded as under the *strong feature* regime, the performance improvement is significant with respect to KS and F1 metrics while being less significant with respect to AUC. Such improvements are still valuable in FRM applications since a higher F1 score potentially suggests better patterns of the precision-recall (PR) curve, which we plot in figure 3. The PR curve shows the dominant performance of GRANDE against baselines: under various precision levels, the recall of GRANDE surpasses the best baseline (TGAT) by as many as 5.29% in absolute value and 13.4% in relative.

E. Ablation study

We evaluate the following variants of GRANDE over all three datasets to investigate contributions of different constituents:

Reduced version this is the one reported in table I

Without causal pruning in this model variant we retain the full edge adjacency graph without pruning. Which is computationally heavier than the GRANDE architecture

Without time encoding in this model variant we discard the temporal component of GRANDE and use the update rule (9)

Without cross-query attention in this model variant we discard the cross-query attention module (10), and use $\mathsf{CONCAT}(g_{uv}, h_v, h_u)$ as the output embedding for edge (u, v).

With line graph in this model variant, we use the ordinary directed line graph instead of the proposed augmented edge adjacency graph. i.e., we replace $\overline{N_L^+(uv)}$ and $\overline{N_L^-(uv)}$ in (9) with $N_L^+(uv)$ and $N_L^-(uv)$, respectively.

Results we report results in table II using the same training configuration and evaluation metrics as in section IV-C. There are a couple of notable observations: Firstly, the causal pruning procedure saves computation as well as improves performance, providing a solid relational inductive bias in temporal graph modeling. Secondly, the incorporation of time encoding and cross-query attention are in general helpful. Finally, using the ordinary line graph performs on par with the reduced model, showing the insufficiency of additional information provided by line digraphs, thereby verifying the necessity of using the augmented edge adjacency graph.

V. Discussion

Interpretability The dominating performance of neural approaches comes at the cost of lacking of *model interpretablity*, which is crucial to application scenarios like AML, where outputs of decision making systems tie strongly with regulatory

strictures [20]. The adoption of neural approaches enjoys better performance than potentially interpretable methods like linear models as well as losing interpretability. Model explanation methods targeting graph neural models are especially challenging [22] due to the combinatorial nature of the interpretation problem. Off-the-shelf GNN explaining tools (refer to [22] and references therein) are not yet applicable to neural models over directed graphs, which is a promising and challenging direction for future explorations.

Multi-task adaptations The representation quality in both node and edge embeddings gives the GRANDE architecture the possibility to exploit side-information via multi-task learning paradigms [26]. For example it is quite common in FRM scenarios to obtain both a set of user riskiness labels alongside transaction labels. We have taken a trivial adaptation of GRANDE into multi-task setups using extra node labels via adding a node-level classification loss to the training objective and has shown solid improvement (we report results in appendix A). Applying more elegant multi-task learning techniques [26] to further exploit the potential of GRANDE is an interesting direction for future studies.

VI. Conclusion

In this paper we propose a graph representation learning framework for directed multigraphs that prevail in FRM applications. The proposed framework generalizes the acclaimed message passing graph neural network protocol to incorporate directional information, as well as utilizing the edge-to-node dual relationship to further enhance the relational inductive bias with regard to edge property prediction tasks. A concrete architecture named GRANDE is derived according to the proposed protocol with the transformer architecture being its aggregation mechanism, as well as a cross-query attention module targeting edge-type tasks. The GRANDE model is generalizable to temporal dynamic graphs via proper generic time encodings along with a pruning strategy. Experimental results over both public and industrial datasets verify the efficacy of the design of GRANDE .

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (2016), pp. 265–283.

[2] Bang-Jensen, J., and Gutin, G. Z. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[3] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[4] Cai, L., Li, J., Wang, J., and Ji, S. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1.

[5] Chen, Z., Li, X., and Bruna, J. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415* (2017).

[6] Chen, Z., Van Khoa, L. D., Teoh, E. N., Nazir, A., Karuppiah, E. K., and Lam, K. S. Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems 57*, 2 (2018), 245–285.

[7] Chung, F. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics 9*, 1 (2005), 1–19.

[8] Dwivedi, V. P., and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020).

[9] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning* (International Convention Centre, Sydney, Australia, 06–11 Aug 2017), D. Precup and Y. W. Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 1263–1272.

[10] Godsil, C., and Royle, G. F. *Algebraic graph theory*, vol. 207. Springer Science & Business Media, 2001.

[11] Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning 14*, 3 (2020), 1–159.

[12] He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising* (2014), pp. 1–9.

[13] Jiang, X., Zhu, R., Li, S., and Ji, P. Co-embedding of nodes and edges with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1.

[14] Jo, J., Baek, J., Lee, S., Kim, D., Kang, M., and Hwang, S. J. Edge representation learning with hypergraphs. *arXiv preprint arXiv:2106.15845* (2021).

[15] Kazemi, S. M., and Goel, R. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research 21*, 70 (2020), 1–73.

[16] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[17] Krzakala, F., Moore, C., Mossel, E., Neeman, J., Sly, A., Zdeborová, L., and Zhang, P. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences 110*, 52 (2013), 20935–20940.

[18] Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018), ACM, pp. 333–341.

[19] Kumar, S., Spezzano, F., Subrahmanian, V., and Faloutsos, C. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* (2016), IEEE, pp. 221–230.

[20] Kute, D. V., Pradhan, B., Shukla, N., and Alamri, A. Deep learning and explainable artificial intelligence techniques applied for detecting money laundering–a critical review. *IEEE Access 9* (2021), 82300–82317.

[21] Liu, C., Sun, L., Ao, X., Feng, J., He, Q., and Yang, H. Intention-aware heterogeneous graph attention networks for fraud transactions detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021), pp. 3280–3288.

[22] Liu, N., Feng, Q., and Hu, X. Interpretability in graph neural networks. In *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022, pp. 121–147.

[23] Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., and Song, L. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 2077–2085.

[24] Ma, Y., Hao, J., Yang, Y., Li, H., Jin, J., and Chen, G. Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990* (2019).

[25] Mashrur, A., Luo, W., Zaidi, N. A., and Robles-Kelly, A. Machine learning for financial risk management: A survey. *IEEE Access 8* (2020), 203203–203223.

[26] Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[27] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks 20*, 1 (2009), 61–80.

[28] Sudjianto, A., Yuan, M., Kern, D., Nair, S., Zhang, A., and Cela-Díaz, F. Statistical methods for fighting financial crimes. *Technometrics 52*, 1 (2010), 5–19.

[29] Tian, S., Wu, R., Shi, L., Zhu, L., and Xiong, T. Self-supervised representation learning on dynamic graphs. In *Proceedings of the 30th*

ACM International Conference on Information & Knowledge Management (2021), pp. 1814–1823.

[30] TONG, Z., LIANG, Y., SUN, C., LI, X., ROSENBLUM, D., AND LIM, A. Digraph inception convolutional networks. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 17907–17918.

[31] TONG, Z., LIANG, Y., SUN, C., ROSENBLUM, D. S., AND LIM, A. Directed graph convolutional network, 2020.

[32] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[33] VELIČKOVIĆ, P., CUCURULL, G., CASANOVA, A., ROMERO, A., LIO, P., AND BENGIO, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[34] WANG, Z., LEI, Y., AND LI, W. Neighborhood interaction attention network for link prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2019), CIKM '19, Association for Computing Machinery, p. 2153–2156.

[35] WANG, Z., ZHOU, Y., HONG, L., ZOU, Y., SU, H., AND CHEN, S. Pairwise learning for neural link prediction, 2021.

[36] XIONG, T., ZHU, L., WU, R., AND QI, Y. Memory augmented design of graph neural networks, 2021.

[37] XU, D., RUAN, C., KORPEOGLU, E., KUMAR, S., AND ACHAN, K. Self-attention with functional time representation learning. In *Advances in Neural Information Processing Systems* (2019), pp. 15889–15899.

[38] XU, D., RUAN, C., KORPEOGLU, E., KUMAR, S., AND ACHAN, K. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[39] XU, K., HU, W., LESKOVEC, J., AND JEGELKA, S. How powerful are graph neural networks? In *International Conference on Learning Representations* (2019).

[40] YING, C., CAI, T., LUO, S., ZHENG, S., KE, G., HE, D., SHEN, Y., AND LIU, T.-Y. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234* (2021).

[41] YOU, J., YING, Z., AND LESKOVEC, J. Design space for graph neural networks. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 17009–17021.

[42] ZAHEER, M., KOTTUR, S., RAVANBAKHSH, S., POCZOS, B., SALAKHUTDINOV, R. R., AND SMOLA, A. J. Deep sets. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[43] ZHANG, D., HUANG, X., LIU, Z., HU, Z., SONG, X., GE, Z., ZHANG, Z., WANG, L., ZHOU, J., SHUANG, Y., ET AL. Agl: A scalable system for industrial-purpose graph machine learning. *arXiv preprint arXiv:2003.02454* (2020).

[44] ZHANG, X., HE, Y., BRUGNONE, N., PERLMUTTER, M., AND HIRN, M. Magnet: A neural network for directed graphs, 2021.

APPENDIX

We list the summary statistics of used datasets in table III

TABLE III: Summary statistics of the evaluation datasets

	Bitcoin-otc	Bitcoin-alpha	AML
# Nodes	5881	3783	10268164
# Edges	35592	24186	13335278
# Positive edges	3563	1536	1338425
# Negative edges	32029	22650	11996853
# Node features	12012	15210	6400
# Edge features	–	–	6400

We conduct an additional experiment using the AML dataset augmenting with a set of node labels: the labels are obtained via an expert-maintained malicious user list that takes the form of a binary vector indicating whether the user is malicious or not. The labels are mapped separately to both the buyer (the party that sends money) and seller (the party that receives money) of the transactions, so that for each transaction we may have up to three labels. We follow exactly the same training configuration and hyperparameter settings in section IV-C. Comparisons are made under the recall metric under different precisions, which we report in table IV. The results imply that the incorporation of side-information yields consistent improvements, especially in recalls at high precision (with a relative improvement of over 10% in r@p95), which is wildly considered to be a key factor in the assessment of models in FRM scenarios.

TABLE IV: Performance comparison between GRANDE and GRANDE with side-information (node-level labels) over the AML dataset. metric r@pp means recall value at precision p, with best performance in bold

	r@p95	r@p90	r@p85	r@p80
GRANDE	0.160	0.368	0.517	0.624
GRANDE (multi-task)	**0.186**	**0.382**	**0.523**	**0.629**