Provably Powerful Graph Neural Networks for Directed Multigraphs

Béni Egressy[1, 2] Luc von Niederhäusern[1, 2], Jovan Blanuša[2], Erik Altman[3], Roger Wattenhofer[1], Kubilay Atasu[2]
[1]ETH Zurich, Zurich, Switzerland
[2]IBM Research Europe, Zurich, Switzerland
[3]IBM Watson Research, Yorktown Heights, NY, USA
begressy@ethz.ch, lucv@ethz.ch, jov@zurich.ibm.com, ealtman@us.ibm.com, wattenhofer@ethz.ch, kat@zurich.ibm.com

Abstract

This paper analyses a set of simple adaptations that transform standard message-passing Graph Neural Networks (GNN) into provably powerful directed multigraph neural networks. The adaptations include multigraph port numbering, ego IDs, and reverse message passing. We prove that the combination of these theoretically enables the detection of any directed subgraph pattern. To validate the effectiveness of our proposed adaptations in practice, we conduct experiments on synthetic subgraph detection tasks, which demonstrate outstanding performance with almost perfect results.

Moreover, we apply our proposed adaptations to two financial crime analysis tasks. We observe dramatic improvements in detecting money laundering transactions, improving the minority-class F1 score of a standard message-passing GNN by up to 30%, and closely matching or outperforming tree-based and GNN baselines. Similarly impressive results are observed on a real-world phishing detection dataset, boosting three standard GNNs' F1 scores by around 15% and outperforming all baselines. An extended version with appendices can be found on arXiv: https://arxiv.org/abs/2306.11586.

Introduction

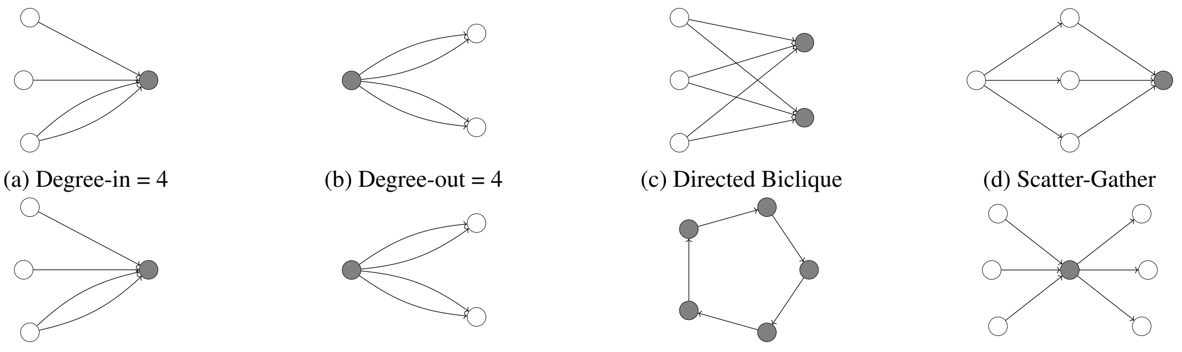
Graph neural networks (GNNs) have become the go-to machine learning models for learning from relational data. GNNs are used in various fields, ranging from biology, physics, and chemistry to social networks, traffic, and weather forecasting (Bongini, Bianchini, and Scarselli 2021; Zhou et al. 2020; Derrow-Pinion et al. 2021; Shu, Wang, and Liu 2019; Wu et al. 2020; Keisler 2022; Zhang et al. 2019; Battaglia et al. 2016). More recently, there has been growing interest in using GNNs to identify financial crime (Cardoso, Saleiro, and Bizarro 2022; Kanezashi et al. 2022; Weber et al. 2019, 2018; Nicholls, Kuppa, and Le-Khac 2021).

Our motivating task is to detect financial crimes manifesting as subgraph patterns in transaction networks. For example, see Fig. 1, which depicts established money laundering patterns. But note that similar patterns are relevant for graph tasks in many areas, ranging from chemistry to traffic forecasting. The task seems to lend itself nicely to the use of GNNs. Unfortunately, current GNNs are ill-equipped to deal with financial transaction networks effectively.

Firstly, financial transaction networks are, in fact, directed multigraphs, i.e., edges (or transactions) have a direction, and there can be multiple edges between two nodes (or accounts). Secondly, most GNNs cannot detect some subgraph patterns, such as cycles (Chen et al. 2020, 2019). There have been many efforts to overcome this limitation (You et al. 2021; Huang et al. 2022; Papp and Wattenhofer 2022; Zhang and Li 2021; Loukas 2019; Sato, Yamada, and Kashima 2019), all focusing on simple (undirected) graphs. But even on simple graphs, the problem is far from solved. Until very recently, for example, there was no linear-time permutation-equivariant GNN that could count 6-cycles (Huang et al. 2022).

This paper addresses both of these issues. To our knowledge, this is the first GNN architecture designed specifically for directed multigraphs. Secondly, we first prove that the proposed architecture can theoretically detect any subgraph pattern in directed multigraphs and then empirically confirm that our proposed architecture can detect the patterns illustrated in Fig. 1. Our proposed architecture is based on a set of simple adaptations that can transform any standard GNN architecture into a directed multigraph GNN. The adaptations are reverse message passing (Jaume et al. 2019), port numbering (Sato, Yamada, and Kashima 2019), and ego IDs (You et al. 2021). Although these individual building blocks are present in existing literature, the theoretical and empirical power of combining them has not been explored. In this work, we fill this gap: We combine them, adapt them to directed multigraphs, and showcase the theoretical and empirical advantages of using them in unison.

Our contributions. (1) We propose a set of simple and intuitive adaptations that can transform message-passing GNNs into provably powerful directed multigraph neural networks. (2) We prove that suitably powerful GNNs equipped with ego IDs, port numbering, and reverse message passing can identify any directed subgraph pattern. (3) The theory is tested on synthetic graphs, confirming that GNNs using these adaptations can detect a variety of subgraph patterns, including directed cycles up to length six, scatter-gather patterns, and directed bicliques, setting them apart from previous GNN architectures. (4) The improvements translate to significant gains on two financial datasets. The adaptations boost GNN performance dramatically on money laundering and phishing datasets, matching or surpassing state-of-the-art financial crime detection models on both simulated and real data.


(a) Degree-in = 4 (b) Degree-out = 4 (c) Directed Biclique (d) Scatter-Gather

(e) Fan-in = 3 (f) Fan-out = 2 (g) Directed Cycle (5) (h) Gather-Scatter

Figure 1: Money Laundering Patterns. The gray fill indicates the nodes to be detected by the synthetic pattern detection tasks. The exact degree/fan pattern sizes here are for illustrative purposes only.

Related Work

Xu et al. (2018) showed that standard MPNNs are at most as powerful as the Weisfeiler-Lehman (WL) isomorphism test, and provided a GNN architecture, GIN, that theoretically matches the power of the WL test. Although the WL test can asymptotically almost surely differentiate any two non-isomorphic graphs (Babai, Erdos, and Selkow 1980), standard MPNNs cannot — in certain graphs — detect simple substructures like cycles (Chen et al. 2020, 2019). This motivated researchers to go beyond standard MPNNs.

One direction considers emulating the more powerful k-WL isomorphism test, by conducting message passing between k-tuples or using a tensor-based model (Maron et al. 2019; Morris et al. 2019). Unfortunately, these models have high complexity and are impractical for most applications. Another line of work uses pre-calculated features to augment the GNN. These works explore adding subgraph counts (Bouritsas et al. 2022; Barceló et al. 2021), positional node embeddings (Egressy and Wattenhofer 2022; Dwivedi et al. 2021), random IDs (Abboud et al. 2020; Sato, Yamada, and Kashima 2021), and node IDs (Loukas 2019).

A recent class of expressive GNNs called Subgraph GNNs, model graphs as collections of subgraphs (Frasca et al. 2022; Zhao et al. 2021). Papp et al. (2021) drop random nodes from the input and run the GNN multiple times, gathering more information with each run. Zhang and Li (2021) instead extract subgraphs around each node and run the GNN on these. Also falling into this category is ID-GNN, which uses ego IDs (You et al. 2021), whereby each node is sampled with its neighborhood and given an identifier to differentiate it from the neighbors. Although the authors claim that ID-GNNs can count cycles, the proof turns out to be incorrect. In fact, Huang et al. (2022) show that the whole family of Subgraph GNNs cannot count cycles of length greater than 4, and propose I^2-GNNs that can count cycles up to length 6.

There has been much less work on GNNs for directed graphs. Zhang et al. (2021) propose a spectral network for directed graphs, but it is difficult to analyze the power of this network or apply it to larger datasets. Similar approaches can be found in (Tong et al. 2020) and (Ma et al. 2019). Jaume et al. (2019) extend message passing to aggregate incoming and outgoing neighbors separately, rather than naively treating the graph as undirected. Directed multigraphs have not specifically been considered.

GNNs have been used for various financial applications (Li et al. 2021; Feng et al. 2019; Chen, Wei, and Huang 2018; Zhang et al. 2019; Li et al. 2019; Xu et al. 2021; Yang et al. 2021). Closest to our work, GNNs have been used for fraud detection. Liang et al. (2019) and Rao et al. (2021) work on bipartite customer-product graphs to uncover insurance and credit card fraud, respectively. Liu et al. (2018) use heterogeneous GNNs to detect malicious accounts in the device-activity bipartite graph of an online payment platform. Weber et al. (2019) were the first to apply standard GNNs for anti-money laundering (AML), and more recently Cardoso, Saleiro, and Bizarro (2022) proposed representing the transaction network as a bipartite account-transaction graph and showed promising results in the semi-supervised AML setting. However, it is not clear how these approaches help with detecting typical fraud patterns.

Background

Graphs and Financial Transaction Graphs

We consider directed multigraphs, G, where the nodes $v \in V(G)$ represent accounts, and the directed edges $e = (u, v) \in E(G)$ represent transactions from u to v. Each node u (optionally) has a set of account features $h^{(0)}(u)$; this could include the account number, bank ID, and account balance. Each transaction $e = (u, v)$ has a set of associated transaction features $h^{(0)}_{(u,v)}$; this includes the amount, currency, and timestamp of the transaction. The incoming and outgoing neighbors of u are denoted by $N_{in}(u)$ and $N_{out}(u)$ respectively. Multiple transactions between the same two accounts are possible, making G a multigraph. In node (or edge) prediction tasks, each node (or edge) will have a binary label indicating whether the account (or transaction) is illicit.

Financial Crime Patterns. Fig. 1 shows a selection of subgraph patterns indicative of money laundering (Granados and Vargas 2022; He et al. 2021; Suzumura 2022; Weber et al. 2018; Starnini et al. 2021). Unfortunately, these are rather generic patterns, which also appear extensively amongst perfectly innocent transactions. As a result, detecting financial crime relies not just on detecting individual patterns, but also on learning relevant combinations. This makes neural networks promising candidates for the task. However, standard message-passing GNNs typically fail to detect the depicted patterns, except for degree-in. In the next section, we describe architectural adaptations, which enable GNNs to detect each one of these patterns.

Subgraph Detection. Given a subgraph pattern H, we define subgraph detection for nodes as deciding for each node in a graph whether it is part of a subgraph that is isomorphic to H; i.e., given a node $v \in V(G)$, deciding whether there exists a graph G', with $E(G') \subseteq E(G)$ and $V(G') \subseteq V(G)$, such that $v \in V(G')$ and $G' \cong H$.

Message Passing Neural Networks

Message-passing GNNs, commonly referred to as Message Passing Neural Networks (MPNNs), form the most prominent family of GNNs. They include GCN (Kipf and Welling 2016), GIN (Xu et al. 2018), GAT (Veličković et al. 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and many more architectures. They work in three steps: (1) Each node sends a message with its current state $h(v)$ to its neighbors, (2) Each node aggregates all the messages it receives from its neighbors in the embedding $a(v)$, and (3) Each node updates its state based on $h(v)$ and $a(v)$ to produce a new state. These 3 steps constitute a layer of the GNN, and they can be repeated to gather information from further and further reaches of the graph. More formally:

$$a^{(t)}(v) = \textsc{Aggregate}\left(\{h^{(t-1)}(u) \mid u \in N(v)\}\right),$$
$$h^{(t)}(v) = \textsc{Update}\left(h^{(t-1)}(v), a^{(t)}(v)\right),$$

where $\{\{.\}\}$ denotes a multiset, and $\textsc{Aggregate}$ is a permutation-invariant function. We will shorten $\textsc{Aggregate}$ to $\textsc{Agg}$, and for readability, we will use $\{.\}$ rather than $\{\{.\}\}$ to indicate multisets.

In the case of directed graphs, we need to distinguish between the incoming and outgoing neighbors of node u. In a standard MPNN, the messages are passed along the directed edges in the direction indicated. As such, the aggregation step only considers messages from incoming neighbors:

$$a^{(t)}(v) = \textsc{Agg}\left(\{h^{(t-1)}(u) \mid u \in N_{in}(v)\}\right),$$

where we aggregate over the incoming neighbors, $N_{in}(v)$.

The edges of an input graph may also have input features. We denote the input features of directed edge $e = (u, v)$ by $h^{(0)}((u, v))$. When using edge features during the message passing, the aggregation step becomes:

$$a^{(t)}(v) = \textsc{Agg}\left(\{(h^{(t-1)}(u), h^{(0)}((u, v))) \mid u \in N_{in}(v)\}\right)$$

In the remainder, we omit edge features from formulas when unnecessary in favor of brevity.

Methods

In this section, we introduce simple adaptations for standard MPNNs (Message Passing Neural Networks) to enable the detection of the fraud patterns in Fig. 1. We consider the adaptations in increasing order of complexity in terms of the patterns they help to detect. We provide theory results to motivate the adaptations and include corresponding experiments on the synthetic subgraph detection dataset in the *Results* section to support the theory empirically.

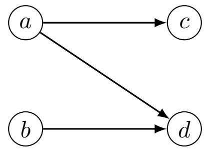


Figure 2: Nodes (a and b) with different out-degrees are not distinguishable by a standard MPNN with directed message passing. Note that naive bidirectional message passing, on the other hand, is unable to distinguish nodes a and d.

Reverse Message Passing

When using a standard MPNN with directed edges, a node does not receive any messages from outgoing neighbors (unless they happen to also be incoming neighbors), and so is unable to count its outgoing edges. For example, a standard MPNN is unable to distinguish nodes a and b in Fig. 2. Further, note that naive bidirectional message passing, where edges are treated as undirected and messages travel in both directions, does not solve the problem, because a node then can not distinguish incoming and outgoing edges. So this would fail to distinguish nodes a and d in the same figure.

To overcome this issue, we need to indicate the direction of the edges in some way. We propose using a separate message-passing layer for the incoming and outgoing edges respectively, i.e., adding *reverse message passing*. Note that this is akin to using a relational GNN with two edge types (Schlichtkrull et al. 2018). More formally, the aggregation and update mechanisms become:

$$a_{in}^{(t)}(v) = \textsc{Agg}_{in}\left(\{h^{(t-1)}(u) \mid u \in N_{in}(v)\}\right),$$
$$a_{out}^{(t)}(v) = \textsc{Agg}_{out}\left(\{h^{(t-1)}(u) \mid u \in N_{out}(v)\}\right),$$
$$h^{(t)}(v) = \textsc{Update}\left(h^{(t-1)}(v), a_{in}^{(t)}(v), a_{out}^{(t)}(v)\right),$$

where a_{in} is now an aggregation of incoming neighbors and a_{out} of outgoing neighbors. We now prove that message-passing GNNs with reverse MP can solve degree-out.

Proposition 0.1. *An MPNN with sum aggregation and reverse MP can solve degree-out.*

The proof of Proposition 0.1 can be found in the appendix. We use a synthetic pattern detection task later in the paper to confirm that the theory translates into practice.

Directed Multigraph Port Numbering

People often make multiple transactions to the same account. In transaction networks, these are represented as parallel edges. To detect fan-in (or fan-out) patterns, a model has to distinguish between edges from the same neighbor and edges

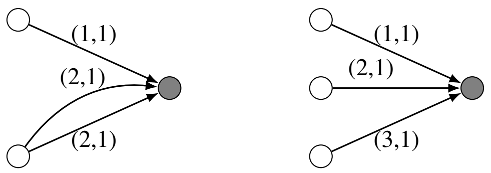


Figure 3: Nodes (in gray) with different fan-ins that are not distinguishable by a standard MPNN. The edge labels indicate incoming and outgoing port numbers, respectively.

from different neighbors. Using unique account numbers (or in general node IDs) would naturally allow for this. However, using account numbers does not generalize well. During training, a model can memorize fraudulent account numbers without learning to identify fraudulent patterns, but this will not generalize to unseen accounts.

Instead, we adapt port numbering (Sato, Yamada, and Kashima 2019) to directed multigraphs. Port numbering assigns local IDs to each neighbor at a node. This allows a node to identify messages coming from the same neighbor in consecutive message-passing rounds. To adapt port numbering to directed multigraphs, we assign each directed edge an incoming and an outgoing port number, and edges coming from (or going to) the same node, receive the same incoming (or outgoing) port number. Unlike Sato, Yamada, and Kashima (2019), who attach only the local port numbers at a node to received messages, we attach the port numbers in both directions, i.e., a node sees both the port number it has assigned to a neighbor and the port number that the neighbor has assigned to it. This turns out to be crucial for our expressivity arguments.

Port numbers have been shown to increase the expressivity of GNNs on simple graphs, but message-passing GNNs with port numbers alone cannot even detect 3-cycles in some cases (Garg, Jegelka, and Jaakkola 2020).

In general, the assignment of port numbers around a node is arbitrary. A node with d incoming neighbors can assign incoming port numbers in $d!$ ways. To break this symmetry in our datasets, we use the transaction timestamps to order the incoming (or outgoing) neighbors. In the case of parallel edges, we use the earliest timestamp to decide the order of the neighbors. Since timestamps carry meaning in financial crime detection, the choice of ordering is motivated; indeed two identical subgraph patterns with different timestamps can have different meanings.

Computing the port numbers in this way can be a time-intensive step, with runtime complexity dominated by sorting all edges by their timestamps: $\mathcal{O}(m \log m)$, where $m = |E(G)|$. However, port numbers can be calculated in advance, so training and inference times are unaffected. Each edge receives an incoming and an outgoing port number as additional edge features. Fig. 3 shows an example of graphs with port numbers. We now prove that GNNs using port numbers can correctly identify fan-in and fan-out patterns.

Note that the following proof, and later proofs using port numbers, do not rely on the timestamps for correctness. However, if timestamps that uniquely identify the ports are available, then permutation invariance/equivariance of the GNN will be preserved.

Proposition 0.2. *An MPNN with max aggregation and multigraph port numbering can solve fan-in.*

A proof is provided in the appendix. Adding reverse MP, one can argue similarly that fan-out can also be solved. Both propositions are confirmed empirically in the *Results* section.

Proposition 0.3. *An MPNN with max aggregation, multigraph port numbering, and reverse MP can solve fan-out.*

Ego IDs

Although reverse MP and multigraph port numbering help with detecting some of the suspicious patterns in Fig. 1, they are not sufficient to detect directed cycles, scatter-gather patterns, and directed bicliques. You et al. (2021) introduced ego IDs specifically to help detect cycles in graphs. The idea is that by "marking" a "center" node with a distinct (binary) feature, this node can recognize when a sequence of messages cycles back around to it, thereby detecting cycles that it is part of. However, it turns out that the proof of Proposition 2 in the paper is incorrect, and ego IDs alone do not enable cycle detection. We give a counterexample in the appendix. Indeed, Huang et al. (2022) also note that the proof "confuses walks with paths".

We see this reflected in the individual results in Table 1. Although ego IDs offer a boost in detecting short cycles, they do not help the baseline (GIN) in detecting longer cycles. This can also be explained theoretically: Assuming a graph has no loops (edges from a node to itself), walks of length two and three that return to the start node are also cycles since there is no possibility to repeat intermediate nodes. Therefore Proposition 2 from You et al. (2021) applies in these cases and it is not surprising that GIN+EgoIDs can achieve impressive F1 scores for 2- and 3-cycle detection.

However, in combination with reverse MP and port numbering, ego IDs can detect cycles, scatter-gather patterns, and bipartite subgraphs, completing the list of suspicious patterns. In fact, it can be shown that a suitably powerful standard MPNN with these adaptations can distinguish any two non-isomorphic (sub-)graphs, and given a *consistent* use of port-numbering they will not mistakenly distinguish any two isomorphic (sub-)graphs. GNNs fulfilling these two properties are often referred to as *universal*. The crux of the proof is showing how the ego ID, port numbers, and reverse MP can be used to assign unique IDs to each node in the graph. Given unique node IDs, sufficiently powerful standard MPNNs are known to be universal (Loukas 2019; Abboud et al. 2020).

Theorem 0.4. *Ego IDs combined with port numbering and reverse MP can be used to assign unique node IDs in connected directed multigraphs.*

The idea of the proof is to show how a GNN can replicate a labeling algorithm that assigns unique IDs to each node in an ego node's neighborhood. The labeling algorithm as well as the full proof are provided in the appendix. The universality of the adaptations follows from this theorem.

Corollary 0.4.1. *GIN with ego IDs, port numbering, and reverse MP can theoretically detect any directed subgraph pattern.*

The proof follows from Theorem 0.4 above and Corollary 3.1 from Loukas (2019). Similar statements can be made for simple undirected graphs. One can remove the reverse MP from the assumptions since this is only needed to make the proof work with directed edges.

Theorem 0.5. *Ego IDs and port numbering can be used to assign unique node IDs in connected undirected graphs.*

Corollary 0.5.1. *GIN with ego IDs and port numbering can theoretically detect any subgraph in undirected graphs.*

The ablation study in Table 1 of the results again supports the theoretical analysis. The combination of the three adaptations achieves impressive scores for all subgraph patterns.

Note that passing the port numbers of both incident nodes of an edge is crucial for inferring unique node IDs. We illustrate this with a simple example in the appendix. In particular, port numbering, as introduced by Sato, Yamada, and Kashima (2019), is not sufficient.

Complexity & Runtime

We propose a set of adaptations, so the final model complexity will depend on the choice of base GNN. We describe the additional runtime costs incurred by the adaptations in the appendix. All in all, the adaptations add a constant factor to the runtime complexity in addition to a one-off pre-computation cost of $\mathcal{O}(m\log(m))$. The empirical runtimes on AML Small HI using GIN can be seen in in the appendix.

Datasets

Synthetic Pattern Detection Tasks. The AML subgraph patterns seen in Fig. 1 are used to create a controllable testbed of synthetic pattern detection tasks. The key design principle is to ensure that the desired subgraph patterns appear randomly, rather than being inserted post hoc into a graph. The problem with inserting patterns is that it skews the random distribution, and simple indicators (such as the degrees of nodes) can be enough to solve the task approximately. For example, consider the extreme case of generating a random k-regular graph and then inserting a pattern. Nodes belonging to the pattern could be identified by checking whether their degree exceeds k. Additionally, if only inserted patterns are labeled, then randomly occurring patterns will be overlooked.

To ensure that the desired subgraph patterns appear randomly, we introduce the *random circulant graph* generator. Details of the generator and pseudocode can be found in the appendix. The pattern detection tasks include degree-in/out (number of in/out edges), fan-in/out (number of unique in/out neighbors), scatter-gather, directed biclique, and directed cycles of length up to six. Detailed descriptions can be found in the appendix.

Anti-Money Laundering (AML). Given the strict privacy regulations around financial data, real-world datasets are not readily available. Instead, we use simulated money laundering data (Altman et al. 2023). The simulator behind these datasets generates a financial transaction network by modeling agents (banks, companies, and individuals) in a virtual world. The generator uses well-established laundering patterns to add realistic money laundering (illicit) transactions. We use two small and two medium-sized datasets, one of each with a higher illicit ratio (HI) and with a lower illicit ratio (LI). The dataset sizes and illicit ratios are provided in the appendix. We use a 60-20-20 temporal train-validation-test split, i.e., we split the transactions after ordering them by their timestamps. Details can be found in the appendix.

Ethereum Phishing Detection (ETH). Since banks do not release their data, we turn to cryptocurrencies for a real-world dataset. We use an Ethereum transaction network published on Kaggle (Chen et al. 2021), where some nodes are labeled as phishing accounts. We use a temporal train-validation-test split, but this time splitting the nodes. We use a 65-15-20 split because the illicit accounts are skewed towards the end of the dataset. More details and dataset statistics can be found in the appendix.

Real-World Directed Graph Datasets. The theory results and the subgraph detection tasks demonstrate the general purpose potential of the architectural adaptations. However, testing our model on real-world benchmark datasets is important to further support these claims. For lack of established directed multi-graph benchmarks, we have taken three directed graph datasets, Chameleon, Squirrel (Pei et al. 2020), and Arxiv-Year (Hu et al. 2020), and compare our approach with the state-of-the-art model for these benchmarks (Rusch et al. 2022). As these datasets are not the focus of this paper, we leave the experimental details and results to the appendix; please see Appendix G.

Experimental Setup

Base GNNs and Baselines. GIN with edge features (Hu et al. 2019) is used as the main GNN base model with our adaptations added on top. GAT (Veličković et al. 2017) and PNA (Velickovic et al. 2019) are also used as base models, and we refer to their adapted versions as Multi-GAT and Multi-PNA, respectively. All three are also considered baselines. Additionally, GIN with ego IDs can be considered an ID-GNN (You et al. 2021) baseline, and GIN with port numbering can be considered a CPNGNN (Sato, Yamada, and Kashima 2019) baseline. Since AML is an edge classification problem, we also include a baseline using edge updates (Battaglia et al. 2018), denoted GIN+EU. This approach is similar to replacing edges with nodes and running a GNN on said *line graph*, which recently achieved state-of-the-art (SOTA) results in self-supervised money laundering detection (Cardoso, Saleiro, and Bizarro 2022). We also include R-GCN (Schlichtkrull et al. 2018) as a baseline. We do not focus on including a more expansive range of GNN baselines, for the simple reason that without (the proposed) adaptations, they are not equipped to deal with directed multigraphs. However, some additional results with "more expressive" GNNs can be found in the appendix. As far as we are aware, there are no other GNNs that one could expect to achieve SOTA results on directed multigraphs.

We include a baseline representing the parallel line of work in financial crime detection that uses pre-calculated graph-based features (GFs) and tree-based classifiers to classify nodes or edges individually. We train XGBoost (Chen and Guestrin 2016) and LightGBM (Ke et al. 2017) models on the individual edges (or nodes) using the original raw features combined with additional graph-based features. This approach has produced SOTA results in financial applications (Weber et al. 2019; Lo, Layeghy, and Portmann 2022).

Given the size of the AML and ETH datasets, we use neighborhood sampling (Hamilton, Ying, and Leskovec 2017) for all GNN-based models. Further details of the experimental setup for the different datasets can be found in the appendix.

Scoring. Since we have very imbalanced datasets, accuracy and other popular metrics are not suitable. Instead, we use the minority class F1 score. This aligns well with what banks and regulators use in real-world scenarios.

Results

Synthetic Pattern Detection Results

The synthetic pattern detection results can be seen in Table 1. The degree-out results reveal that the standard message-passing GNNs are unable to solve the degree-out task, achieving F1 scores below 44%. However, all the GNNs that are equipped with reverse MP score above 98%, thus supporting Proposition 0.1. The next column shows that port numbering is the critical adaptation for solving fan-in, though the F1 score is quite high even for the baseline GIN. On the other hand, for the fan-out task, the combination of reverse MP and port numbering is needed to score above 99%. Again, these results support Propositions 0.2 and 0.3. The ablation study of cumulative adaptations on top of GIN also supports Corollary 0.4.1: The combination of reverse MP, port numbering, and ego IDs, scores high on all of the subtasks, with only 6-cycle detection coming in below 90%. We see similar results when using other base GNN models, with Multi-PNA achieving the best overall results. Moreover, on the more complex tasks — directed cycle, scatter-gather, and biclique detection — the combination of the three is what leads to the first significant improvement in F1 scores. In the most extreme case, scatter-gather detection, the minority class F1 score jumps from 67.84% with only reverse MP and port numbers to 97.42% when ego IDs are added. No adaptation alone comes close to this score, so it is clear that the combination is needed. Similar jumps can be seen for directed 4-, 5-, and 6-cycle, and biclique detection. Increasing the dataset size and restricting the task to only the "complex" subtasks further increases the scores, with 6-cycle detection also reaching above 97%. More details can be found in the appendix, along with additional ablations. In particular, we rerun the experiments using random unique node IDs as input features and see that node IDs are unable to replace port numbers and ego IDs in practice.

AML Results

The results for the AML datasets can be seen in Table 2. For AML Small HI, we see that our adaptations boost the minority class F1 score of GIN from 28.7% to 57.2%, a gain of almost 30%. The largest improvements are brought by reverse MP and port numbering, taking the F1 score from 28.7% to 56.9%, whilst ego IDs do not make much difference here. The results for the other AML datasets show a similar trend with overall gains of 14.2%, 14.0%, and 10.7% for GIN, again with diminishing returns as more adaptations are added. The two rows corresponding to port numbering — *GIN+Ports* and *+Ports* — indicate clear gains from using port numbering, both when used alone and on top of reverse MP. The support for ego IDs is less clear, with clear gains when used as an individual adaptation but no significant gains when added on top of reverse MP and port numbering.

The full set of adaptations was tested with three other base models, GIN+EU (GIN with edge updates), PNA, and PNA+EU. In each case, and across almost all AML datasets, we see clear gains from using the adaptations, underlining the effectiveness and versatility of the approach. We further note that Multi-PNA+EU outperforms all baselines on all AML datasets. This is particularly impressive when compared with the tree-based methods using graph-based features (XGBoost+GFs and LightGBM+GFs) since the hand-crafted features align perfectly with the illicit money laundering patterns used by the simulator. Moreover, these tree-based methods have been SOTA in previous financial applications (Weber et al. 2019; Lo, Layeghy, and Portmann 2022).

Recall scores for individual money laundering patterns can be found in the appendix. It is worth noting that the majority of the illicit transactions that belong to money laundering patterns are identified, and the overall dataset scores are greatly influenced by the proportion of lone (not belonging to a money laundering pattern) illicit transactions in the datasets. Lone illicit transactions are very difficult to identify.

For training times and inference throughput rates of models based on GIN, please see the appendix. Notably, with all the adaptations, the inference rate of Multi-GIN still surpasses 18k transactions per second on a single GPU.

ETH Results

Finally, we test our adaptations on a real-world financial crime dataset — Ethereum phishing account classification. The results are provided in Table 2. Similar to the AML datasets, we see a consistent improvement in final scores as we add the adaptations. In total, the minority class F1 score jumps from 26.9% without adaptations to 42.9% with reverse MP, port numbering, and ego IDs. Again, the largest single improvement is due to the reverse MP. In this case, Multi-GIN does not outperform all of the baselines, but the adaptations also significantly boost PNA performance, and Multi-PNA and Multi-PNA+EU beat all the baselines by more than 12%.

Conclusion

This work has investigated a series of straightforward adaptations capable of transforming conventional message-passing GNNs into provably powerful directed multigraph learners. Our contributions to the field of graph neural networks are threefold. Firstly, our theoretical analysis addresses a notable gap in the existing literature about the power of combining different GNN adaptations/augmentations. Specifically, we prove that ego IDs combined with port numbering and reverse message passing enable a suitably powerful message-passing GNN, such as GIN, to compute unique node IDs and therefore detect any directed subgraph patterns. Secondly, our theoretical findings are validated empirically with a range

Model	deg-in	deg-out	fan-in	fan-out	C2	C3	C4	C5	C6	S-G	B-C
GIN (Xu et al. 2018; Hu et al. 2019)	**99.77**	43.58	95.57	35.91	34.67	58.00	50.80	43.12	48.59	69.31	63.12
GAT (Veličković et al. 2017)	10.33	10.53	9.69	0.00	0.00	0.00	25.86	0.00	0.00	0.00	0.00
PNA (Velickovic et al. 2019)	**99.63**	43.02	95.00	38.93	25.77	54.75	51.92	48.79	48.40	65.88	65.51
GIN+EU (Battaglia et al. 2018)	**99.30**	42.74	95.70	39.13	32.58	55.91	54.65	47.62	49.68	68.54	64.64
GIN+EgoIDs (You et al. 2021)	**99.78**	51.48	95.06	49.24	**98.13**	**97.97**	53.12	44.37	45.42	66.44	63.90
GIN+Ports (Sato, Yamada, and Kashima 2019)	**99.47**	45.00	**99.59**	41.51	27.79	56.11	42.68	41.11	44.99	67.99	65.76
GIN+ReverseMP (Jaume et al. 2019)	**98.87**	**99.08**	94.99	95.25	35.96	63.85	69.09	67.44	71.23	65.83	66.18
+Ports	**98.41**	**98.35**	**98.51**	**99.16**	39.15	63.58	69.00	70.35	75.04	67.84	65.78
+EgoIDs (Multi-GIN)	**99.48**	**99.09**	**99.62**	**99.32**	**98.97**	**98.73**	97.46	91.60	84.23	**97.42**	94.33
Multi-GAT	**98.68**	**98.36**	**99.28**	**99.33**	98.61	**98.93**	**98.90**	**95.82**	**91.81**	96.66	86.92
Multi-PNA	**99.64**	**99.25**	**99.53**	**99.41**	**99.71**	**99.54**	**99.49**	**97.46**	88.75	**99.07**	**96.77**
Multi-GIN+EU	**99.55**	**99.53**	**99.76**	**99.77**	**99.37**	**99.71**	**98.73**	**95.73**	88.38	**98.81**	**97.82**

Table 1: Minority class F1 scores (%) for the synthetic subgraph detection tasks. First from the top are the standard MPNN baselines; then the results with each adaptation added separately to GIN; followed by GIN with the adaptations added cumulatively; and finally, results for other GNN baselines with the adaptations (Multi-GNNs). The Ck abbreviations stand for directed k-cycle detection, S-G stands for scatter-gather and B-C stands for biclique detection. We report minority class F1 scores averaged over five runs. We omit standard deviations in favor of readability. Scores within 2% of the top score (underlined) are shown in bold.

Model	AML Small HI	AML Small LI	AML Medium HI	AML Medium LI	ETH
LightGBM+GFs (Altman et al. 2023)	62.86 ± 0.25	20.83 ± 1.50	59.48 ± 0.15	20.85 ± 0.38	53.20 ± 0.60
XGBoost+GFs (Altman et al. 2023)	63.23 ± 0.17	27.30 ± 0.33	**65.70 ± 0.26**	28.16 ± 0.14	49.40 ± 0.54
GIN (Xu et al. 2018; Hu et al. 2019)	28.70 ± 1.13	7.90 ± 2.78	42.20 ± 0.44	3.86 ± 3.62	26.92 ± 7.52
PNA (Velickovic et al. 2019)	56.77 ± 2.41	14.85 ± 1.46	59.71 ± 1.91	27.73 ± 1.65	51.49 ± 4.26
GIN+EU (Battaglia et al. 2018)	47.73 ± 7.86	20.62 ± 2.41	49.26 ± 4.02	6.19 ± 8.32	33.92 ± 7.34
R-GCN (Schlichtkrull et al. 2018)	41.78 ± 0.48	7.43 ± 0.38	OOM	OOM	OOM
GIN+EgoIDs (You et al. 2021)	39.65 ± 4.73	14.98 ± 2.66	45.26 ± 2.16	11.17 ± 6.41	26.01 ± 2.27
GIN+Ports (Sato, Yamada, and Kashima 2019)	54.85 ± 0.89	21.41 ± 2.40	54.22 ± 1.94	10.51 ± 12.82	32.96 ± 0.25
GIN+ReverseMP (Jaume et al. 2019)	46.79 ± 4.97	15.98 ± 4.39	51.93 ± 2.90	14.00 ± 9.34	36.86 ± 8.12
+Ports	56.85 ± 2.64	23.80 ± 4.07	57.15 ± 0.76	11.39 ± 8.36	42.51 ± 7.16
+EgoIDs (Multi-GIN)	57.15 ± 4.99	22.12 ± 2.88	56.23 ± 1.51	14.55 ± 2.91	42.86 ± 2.53
Multi-GIN+EU	64.79 ± 1.22	26.88 ± 6.63	58.92 ± 1.83	16.30 ± 4.73	48.37 ± 6.62
Multi-PNA	64.59 ± 3.60	**30.65 ± 2.00**	**65.67 ± 2.66**	33.23 ± 1.31	**65.28 ± 2.89**
Multi-PNA+EU	**68.16 ± 2.65**	**33.07 ± 2.63**	**66.48 ± 1.63**	**36.07 ± 1.17**	**66.58 ± 1.60**

Table 2: Minority class F1 scores (%) for the AML and ETH tasks. HI indicates a higher illicit ratio and LI indicates a lower illicit ratio. The models are organized as in Table 1. "OOM" indicates that the model ran out of GPU memory. Scores within one standard deviation of the top score (underlined) are shown in bold.

of synthetic subgraph detection tasks. The practical results closely mirror the theoretical expectations, confirming that the combination of all three adaptations is needed to detect the more complex subgraphs. Lastly, we show how our adaptations can be applied to two important financial crime problems: detecting money laundering transactions and phishing accounts. GNNs enhanced with our proposed adaptations achieve impressive results in both tasks, either matching or surpassing relevant baselines. Reverse message passing and port numbering again prove crucial in reaching the highest scores, however, we find that ego IDs do not provide much additional benefit for these datasets.

Although this work has focused on financial crime applications, the theory and practical results have a broader relevance. Immediate future work could involve exploring applications of our methods to other directed multigraph problems. An initial exploration can be found in the appendix, showing promising results on three real-world datasets. However, further experiments are needed to confirm general applicability in various domains. Additionally, future work could explore the relationship between the computational complexity of different subgraph detection problems and GNN performance.

Acknowledgements

The support of the Swiss National Science Foundation (project numbers: 172610 and 212158) for this work is gratefully acknowledged.

References

Abboud, R.; Ceylan, I. I.; Grohe, M.; and Lukasiewicz, T. 2020. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*.

Altman, E.; Blanuša, J.; Von Niederhäusern, L.; Egressy, B.; Anghel, A.; and Atasu, K. 2023. Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Babai, L.; Erdos, P.; and Selkow, S. M. 1980. Random graph isomorphism. *SIaM Journal on computing*, 9(3): 628–635.

Barceló, P.; Geerts, F.; Reutter, J.; and Ryschkov, M. 2021. Graph neural networks with local graph parameters. *Advances in Neural Information Processing Systems*, 34: 25280–25293.

Battaglia, P.; Pascanu, R.; Lai, M.; Jimenez Rezende, D.; et al. 2016. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29.

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. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bongini, P.; Bianchini, M.; and Scarselli, F. 2021. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450: 242–252.

Bouritsas, G.; Frasca, F.; Zafeiriou, S. P.; and Bronstein, M. 2022. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cardoso, M.; Saleiro, P.; and Bizarro, P. 2022. LaundroGraph: Self-Supervised Graph Representation Learning for Anti-Money Laundering. In *Proceedings of the Third ACM International Conference on AI in Finance*, 130–138.

Chen, L.; Peng, J.; Liu, Y.; Li, J.; Xie, F.; and Zheng, Z. 2021. Phishing scams detection in Ethereum transaction network. *ACM Trans. Internet Technol.*, 21(1): 1–16.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Chen, Y.; Wei, Z.; and Huang, X. 2018. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1655–1658.

Chen, Z.; Chen, L.; Villar, S.; and Bruna, J. 2020. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33: 10383–10395.

Chen, Z.; Villar, S.; Chen, L.; and Bruna, J. 2019. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32.

Derrow-Pinion, A.; She, J.; Wong, D.; Lange, O.; Hester, T.; Perez, L.; Nunkesser, M.; Lee, S.; Guo, X.; Wiltshire, B.; et al. 2021. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3767–3776.

Dwivedi, V. P.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*.

Egressy, B.; and Wattenhofer, R. 2022. Graph Neural Networks with Precomputed Node Features. *arXiv preprint arXiv:2206.00637*.

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2): 1–30.

Frasca, F.; Bevilacqua, B.; Bronstein, M. M.; and Maron, H. 2022. Understanding and extending subgraph gnns by rethinking their symmetries. *arXiv preprint arXiv:2206.11140*.

Garg, V.; Jegelka, S.; and Jaakkola, T. 2020. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, 3419–3430. PMLR.

Granados, O. M.; and Vargas, A. 2022. The geometry of suspicious money laundering activities in financial networks. *EPJ Data Science*, 11(1): 6.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

He, J.; Tian, J.; Wu, Y.; Cia, X.; Zhang, K.; Guo, M.; Zheng, H.; Wu, J.; and Ji, Y. 2021. An efficient solution to detect common topologies in money launderings based on coupling and connection. *IEEE Intelligent Systems*, 36(1): 64–74.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Huang, Y.; Peng, X.; Ma, J.; and Zhang, M. 2022. Boosting the Cycle Counting Power of Graph Neural Networks with I^2-GNNs. *arXiv preprint arXiv:2210.13978*.

Jaume, G.; Nguyen, A.-p.; Martínez, M. R.; Thiran, J.-P.; and Gabrani, M. 2019. edGNN: a Simple and Powerful GNN for Directed Labeled Graphs. *arXiv preprint arXiv:1904.08745*.

Kanezashi, H.; Suzumura, T.; Liu, X.; and Hirofuchi, T. 2022. Ethereum Fraud Detection with Heterogeneous Graph Neural Networks. *arXiv preprint arXiv:2203.12363*.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Keisler, R. 2022. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Li, C.; Jia, K.; Shen, D.; Shi, C.-J. R.; and Yang, H. 2019. Hierarchical Representation Learning for Bipartite Graphs. In *IJCAI*, volume 19, 2873–2879.

Li, W.; Bao, R.; Harimoto, K.; Chen, D.; Xu, J.; and Su, Q. 2021. Modeling the stock relation with graph network for overnight stock movement prediction. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, 4541–4547.

Liang, C.; Liu, Z.; Liu, B.; Zhou, J.; Li, X.; Yang, S.; and Qi, Y. 2019. Uncovering insurance fraud conspiracy with network learning. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 1181–1184.

Liu, Z.; Chen, C.; Yang, X.; Zhou, J.; Li, X.; and Song, L. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM international conference on information and knowledge management*, 2077–2085.

Lo, W. W.; Layeghy, S.; and Portmann, M. 2022. Inspection-L: Practical GNN-based money laundering detection system for bitcoin. *arXiv preprint arXiv:2203.10465*.

Loukas, A. 2019. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*.

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

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably powerful graph networks. *Advances in neural information processing systems*, 32.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 4602–4609.

Nicholls, J.; Kuppa, A.; and Le-Khac, N.-A. 2021. Financial cybercrime: A comprehensive survey of deep learning approaches to tackle the evolving financial crime landscape. *Ieee Access*, 9: 163965–163986.

Papp, P. A.; Martinkus, K.; Faber, L.; and Wattenhofer, R. 2021. DropGNN: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34: 21997–22009.

Papp, P. A.; and Wattenhofer, R. 2022. A theoretical comparison of graph neural network extensions. In *International Conference on Machine Learning*, 17323–17345. PMLR.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*.

Rao, S. X.; Zhang, S.; Han, Z.; Zhang, Z.; Min, W.; Chen, Z.; Shan, Y.; Zhao, Y.; and Zhang, C. 2021. xFraud: explainable fraud transaction detection. *Proceedings of the VLDB Endowment*, 15: 427–436.

Rusch, T. K.; Chamberlain, B. P.; Mahoney, M. W.; Bronstein, M. M.; and Mishra, S. 2022. Gradient gating for deep multi-rate learning on graphs. *arXiv preprint arXiv:2210.00513*.

Sato, R.; Yamada, M.; and Kashima, H. 2019. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems*, 32.

Sato, R.; Yamada, M.; and Kashima, H. 2021. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 333–341. SIAM.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *Extended Semantic Web Conference*, 593–607. Springer.

Shu, K.; Wang, S.; and Liu, H. 2019. Beyond news contents: The role of social context for fake news detection. In *Proceedings of the twelfth ACM international conference on web search and data mining*, 312–320.

Starnini, M.; Tsourakakis, C. E.; Zamanipour, M.; Panisson, A.; Allasia, W.; Fornasiero, M.; Puma, L. L.; Ricci, V.; Ronchiadin, S.; Ugrinoska, A.; et al. 2021. Smurf-Based Anti-money Laundering in Time-Evolving Transaction Networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 171–186. Springer.

Suzumura, T. 2022. AMLSIM library wiki. https://github.com/IBM/AMLSim/wiki/Transaction-Model:-Alert-Model. Accessed: 30-11-2022.

Tong, Z.; Liang, Y.; Sun, C.; Li, X.; Rosenblum, D.; and Lim, A. 2020. Digraph inception convolutional networks. *Advances in neural information processing systems*, 33: 17907–17918.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. *ICLR (Poster)*, 2(3): 4.

Weber, M.; Chen, J.; Suzumura, T.; Pareja, A.; Ma, T.; Kanezashi, H.; Kaler, T.; Leiserson, C. E.; and Schardl, T. B. 2018. Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076*.

Weber, M.; Domeniconi, G.; Chen, J.; Weidele, D. K. I.; Bellei, C.; Robinson, T.; and Leiserson, C. E. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*.

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

Xu, B.; Shen, H.; Sun, B.; An, R.; Cao, Q.; and Cheng, X. 2021. Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4537–4545.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yang, S.; Zhang, Z.; Zhou, J.; Wang, Y.; Sun, W.; Zhong, X.; Fang, Y.; Yu, Q.; and Qi, Y. 2021. Financial risk analysis for SMEs with graph-based supply chain mining. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4661–4667.

You, J.; Gomes-Selman, J. M.; Ying, R.; and Leskovec, J. 2021. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 10737–10745.

Zhang, M.; and Li, P. 2021. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34: 15734–15747.

Zhang, S.; Yao, L.; Sun, A.; and Tay, Y. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1): 1–38.

Zhang, X.; He, Y.; Brugnone, N.; Perlmutter, M.; and Hirn, M. 2021. Magnet: A neural network for directed graphs. *Advances in neural information processing systems*, 34: 27003–27015.

Zhao, L.; Jin, W.; Akoglu, L.; and Shah, N. 2021. From stars to subgraphs: Uplifting any GNN with local structure awareness. *arXiv preprint arXiv:2110.03753*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI open*, 1: 57–81.