LAS-GNN: A Graph Neural Network for Temporal Money Laundering Motif Detection

Stan Verlaan
Utrecht University
Utrecht, Netherlands
stan.verlaan@proton.me

Ioana Hulpuș
Utrecht University
Utrecht, Netherlands
i.r.karnstedt-hulpus@uu.nl

Erik Jan van Leeuwen
Utrecht University
Utrecht, Netherlands
e.j.vanleeuwen@uu.nl

Abstract

We enhance Graph Neural Networks (GNNs) for identifying suspicious accounts involved in money laundering patterns. Extending the work of Egressy et al. (AAAI 2024), we propose a novel GNN architecture to detect suspicious subgraph motifs in the weighted temporal networks underlying financial data. Our architecture allows for the indication of edge directionality within a single Aggregator function, element-wise edge weight multiplication, and an LSTM aggregator that can learn from the sequential order of edges imposed by timestamps. The resulting model, LAS-GNN, is based on an inductive learning framework and can generalize across different networks. Experimental results on synthetic networks show that LAS-GNN is robust and can identify basic money laundering motifs to near perfection, outperforming a graph isomorphism network benchmark with edge features.

CCS Concepts

• **Computing methodologies** → **Supervised learning**; **Neural networks**.

Keywords

anti-money laundering, financial networks, temporal motif detection, graph neural networks

ACM Reference Format:
Stan Verlaan, Ioana Hulpuș, and Erik Jan van Leeuwen. 2025. LAS-GNN: A Graph Neural Network for Temporal Money Laundering Motif Detection. In *6th ACM International Conference on AI in Finance (ICAIF '25), November 15–18, 2025, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3768292.3770410

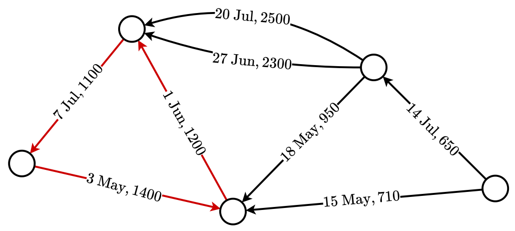


Figure 1: An example of a financial network [1], where the edges include a date and amount of money. Suspicious transactions that together form a cycle are highlighted in red.

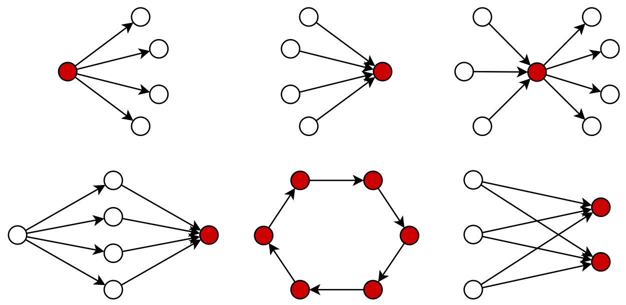


Figure 2: Six (out of eight) money laundering motifs of [43]. *Top row:* fan-out, fan-in, gather-scatter. *Bottow row:* scatter-gather, simple cycle, and bipartite clique. Nodes marked in red can be seen as key accounts and are to be detected.

1 Introduction

Money laundering (ML) is the process of concealing the origin of illegitimately obtained money. Given its association with criminal activities, the detection of ML processes is of paramount importance. Significant progress on machine learning methods for detecting ML activity has been made in recent years (see e.g. the surveys [9, 27, 28]), particularly utilizing the underlying network structure of transaction data [26]. A financial network can be seen as a directed graph, where edges represent transactions between accounts (vertices) and edges can have attributes such as time and value of the transaction [6, 17]. The presence of certain *motifs* or patterns, for example a directed cycle (see Figure 1), can potentially be an indicator of money laundering [16, 19, 42, 46]. Suzumura and Kanezashi [43] introduced a set of common money laundering motifs in financial networks (see Figure 2). For example, the scatter-gather motif is associated with smurfing [42], where a criminal seeks to hide the illicit source of money by splitting it among many accounts, then gathering it before reintroducing it in the economy. Arguably, in ML activities, the chronology of the involved transactions also bears some relevance. Thus, understanding ML activities allows the problem of detecting money laundering processes to be modeled as finding motifs in directed temporal graphs.

Many approaches (machine learning-based and otherwise) exist for finding motifs in directed temporal graphs (see e.g. [4, 5, 17, 24, 30, 33, 39, 49, 54]). We focus on the potential capabilities of Graph Neural Networks (GNNs) [50] for this problem. GNNs learn node representations by taking both the graph structure and the node and edge features into account. Intuitively, they propagate node information along the edges of the graph to update node representations. Combined with multiple layers and non-linear activation functions, this can capture complex relations and patterns in the data and makes GNNs well suited for finding motifs [8, 11, 52]. However, as far as we are aware, no previous GNN-based approach for money laundering motif detection was designed to account for the temporal data in financial transactions.

In recent years, many GNNs that take temporal data into account have been proposed (see the overviews [14, 34, 41, 51]). In contrast to the typical requirements of temporal GNN models, however, the problem of temporal motif detection does not need to compute (and recover) for each node a representation at every time step (or time window). Rather, we aim to represent the nodes based on the chronology of the transactions. The hypothesis is that such chronology-aware node representations are able to identify *key-nodes* that participate in temporal motifs.

We also remark that existing temporal GNN models do not account holistically for edge directions. For example, TGN [37], a GNN framework that generalizes many existing temporal GNNs, learns two different functions for aggregating messages corresponding to incoming and outgoing edges respectively. We show in this paper that such approaches to edge directionality fail to capture the interplay between edge chronology and edge directions.

Therefore, we propose a novel GNN framework for finding temporal and weighted motifs in directed graphs. We build on the approach of Egressy et al. [11], notably their use of ego IDs [53]. To train our model to detect temporal motifs, we employ a long short-term-memory (LSTM) [20] aggregator that learns from the sequential order of edges imposed by timestamps. Moreover, we propose a novel message passing scheme for directed graphs, termed *signed message passing*, that allows for the aggregation of messages from incoming and outgoing neighbours simultaneously.

We evaluate LAS-GNN on synthetically generated networks. Our experiments isolate the effect of each model component: directed message passing, edge weight representation, and the LSTM aggregator. The results show that the components work synergistically, achieving almost perfect detection of the targeted temporal motifs. We also compare its performance at detecting temporal motifs to the state-of-the-art GNN architecture of Egressy et al. [11].

2 Related Work

One can identify two main and complementary directions for detecting money laundering activity in financial networks: (i) data-driven approaches where the model is trained directly on real financial data [7, 10, 31], and (ii) expert-driven pattern matching approaches where the models are specialized for detecting domain expert defined patterns [12, 19, 32, 43]. Our work falls in the latter category.

We cast money laundering detection as a motif detection problem. In this direction, methods such as Flowscope [32] are designed to target one motif. GNN models, however, can be trained to detect arbitrary graph motifs, albeit not without challenges.

First, off-the-shelf GNNs are aces at learning node representations but fail at detecting subgraph motifs. Recently, You et al. [53] proposed and Egressy et al. [11] perfected an elegant approach for detecting motifs that are centered on a node: they set the so-called *ego ID* feature of a node, "the ego", to 1, and to 0 for all nodes in its neighbourhood. When ego IDs are paired with a message passing model that allows messages to travel the edges in both directions, intuitively the diffusion of the "ego's" representation through "ego's" neighbourhood captures the graph structure around the "ego".

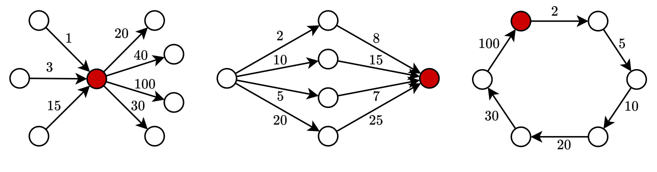
Second, the typical GNN approach to edge directionality is what we call *directional message passing*, where messages strictly follow the direction of the edges. However, in financial networks, outgoing transactions are just as relevant to an account's information as incoming transactions. Simply taking the underlying undirected graph (*bidirectional message passing* [23]) does not differentiate between the sender and the receiver. The most common alternative is to define separate Aggregator functions for the incoming and outgoing edges, respectively [11, 37, 38]. To emphasize that this method implicitly defines two types of relations in the graph, we name it *heterogeneous message passing*. Egressy et al. [11] show that ego IDs and heterogeneous message passing taken together transform any MPNN into a *universal* GNN, that can distinguish any two non-isomorphic (sub-)graphs, while not mistakenly distinguishing any two isomorphic (sub-)graphs. Nevertheless, as we show in this paper, heterogeneous message passing cannot capture the relation between edge directions and edge timestamps.

Lastly, ML activities oftentimes involve multiple accounts that synchronize their actions in time, but this chronological aspect has not been addressed so far in the GNN motif detection literature.

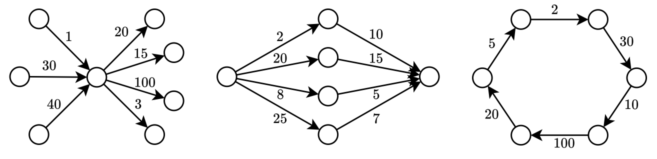
As our work proposes a GNN architecture that we train to detect temporal motifs, this work is also related to the domain of temporal GNNs. The problem of temporal graph representation learning takes as input an initial graph and a series of temporally labeled events such as node interactions, node birth, etc. A core aim of these models is to provide for a node v, its representation corresponding to a certain time t [48, 55]. As such, this problem is significantly different from the problem we address: temporal motif detection. We do not analyze the network as it evolves over time; we process a static version of the network with timestamped edges. Also, we are not interested in maintaining a memory of nodes' states over time; rather, we care to obtain precisely one representation for a node, based on the chronology of the transactions in its subgraph.

In our solution to temporal motif detection, we use a long short-term memory (LSTM) cell for aggregating messages from neighbouring nodes. The idea of using an LSTM aggregator in GNNs is not new. Hamilton et al. [18] were the first to propose it. However, a critical concern in GNN research is to enforce permutation invariance: a node's embeddings should be independent from its neighbourhood ordering. This requirement mostly constrains the choice of Aggregator function. Hamilton et al. exploited the expressivity of an LSTM aggregator, but ensured permutation invariance by repeated neighbour sampling and order randomization.

In contrast, in our work we actually exploit the permutation sensitivity of an LSTM aggregator, because we need it to learn from the temporal sequence of transactions. From this perspective, our approach is among the very few [22] permutation sensitive GNN models in literature. Lastly, we note that most temporal GNNs such as TGN [37] (which generalizes many others), use some form of recurrent neural network (usually RNN, GRU, or LSTM) in their architecture. However, this typically occurs in the Update function to smooth the transition between the hidden layers representations of nodes, or in the Memory function to model the sequence of a



(a) Temporal motif occurrences.



(b) Non-temporal motif occurrences.

Figure 3: Examples of temporal and non-temporal motif occurrences in temporal networks. From right to left: gather-scatter, scatter-gather, directed cycle. The numbers on the edges correspond to timestamps.

node's states over time. Rossi et al. [37] mention the possibility of an RNN aggregator, without further elaboration.

3 Method: LAS-GNN

3.1 Weighted and Temporal Motifs

Given are a (directed) graph $G = (V, E)$ and, for each edge $(u, v) \in E$, a weight $w(u, v) \in [0, 1]$ and an integer timestamp $\text{time}(u, v)$.

Definition 3.1. A (directed) graph H occurs as a *motif* in G if a subgraph H' of G is isomorphic to H; that is, there is a bijection $\phi : V(H') \to V(H)$ such that $(u, v) \in E(H')$ if and only if $(\phi(u), \phi(v)) \in E(H)$. We call ϕ the *isomorphism function.*

We now extend Definition 3.1 to the weighted setting. Edge weights can be interpreted in two ways. First, we can view them as monetary values. Second, we can view them as a money-laundering risk indicator per transaction, set by an external algorithm, e.g. using measures of node relatedness that indicate the strength of the connection. For both cases, having a consistent lower bound on the weight of all edges in the transaction is well motivated; in particular, for the first case, many anti-money laundering directives contain reporting thresholds (see e.g. [13]). Hence, we define:

Definition 3.2. Given $r \in [0, 1]$, a (directed) graph H occurs as a *weighted motif* in G if, for a subgraph H' of G isomorphic to H, it holds that $w(u, v) \geq r$ for all $(u, v) \in H'$.

We now extend Definition 3.1 to the temporal setting. Recall the following well-known graph-theoretic notion. A *feedback node set* (or feedback vertex set) of a (directed) graph H is a set $F \subseteq V(H)$ such that $H - F$ has no (directed) cycle. It is *minimal* if no proper subset of F is also a feedback node set.

Definition 3.3. A (directed) graph H with minimal feedback node set F occurs as a *temporal motif* in G if there is a subgraph H' of G isomorphic to H with isomorphism function ϕ such that for each $v \in \{x \in V(H') \mid \phi(x) \notin F\}$, the inequality $\max_{u \in N_{\text{in}}^{H'}(v)} \{\text{time}(u, v)\} < \min_{w \in N_{\text{out}}^{H'}(v)} \{\text{time}(v, w)\}$ holds.

Informally, the definition states that for every node in the motif, except those in F, the time of all incoming edges in the motif is lower than the time of any outgoing edge in the motif. This generalizes existing definitions of temporal motifs (see e.g. [17, 36, 39]), and allows to consider arbitrary motif graphs. We remark that we do not enforce a time window constraint [36]; rather, we emphasize the chronological aspect of the motif.

Figure 3 illustrates our temporal motif definition. The first two motifs are acyclic, and thus we can use an empty feedback node set; hence, all nodes must respect the time constraint in an occurrence. In the third motif, a cycle, any node can constitute the feedback node set; thus, any one node "is allowed" to violate the constraint. In our example, this is the red node. The red nodes in Figure 3a are the key-nodes we train our model to detect. Our goal is to find temporal motif occurrences such as those illustrated in Figure 3 in a given graph G that occur simultaneously in the weighted (Definition 3.2) and temporal (Definition 3.3) sense. We are approaching the problem as a supervised classification problem, such that the model is trained on a graph that contains these motifs and motif key-nodes are labeled as the positive class, while all other nodes are labeled as the zero class. We are addressing the inductive setting, in which the model must generalize and be able to classify nodes belonging to the targeted motifs in new graphs.

3.2 Model architecture

We propose a novel GNN architecture. Like Egressy et al. [11], our model has ego IDs [53] at its core, but it deviates substantially in its handling of edge directions and time. We describe our key ideas.

First, we propose a single Aggregator function to handle edge directions. We do this by indicating edge direction by weighting the embeddings over the two directions differently within the Aggregator function. As far as we are aware, this approach has not yet appeared in the literature. We call this *signed message passing*:

$$\boldsymbol{a}_v^{(k)} = \textsc{Aggregate}^{(k)}\left(\left\{\boldsymbol{d}_{u,v} \odot \boldsymbol{h}_u^{(k-1)} \mid u \in N(v)\right\}\right) \tag{1}$$

$$\boldsymbol{d}_{u,v} = \begin{cases} \mathit{Linear}(1) & \text{if } u \in N_{\text{in}}(v), \\ \mathit{Linear}(-1) & \text{if } u \in N_{\text{out}}(v), \end{cases} \tag{2}$$

$\textsc{Aggregate}^{(k)}$ is any Aggregator function for layer k. *Linear* is a linear layer that transforms an input scalar into a vector the size of the node embeddings, and $\odot$ represents element-wise multiplication. Instead of multiplying every component of $\boldsymbol{h}_u^{(k-1)}$ with the scalar 1 or −1, the model can learn a separate weight factor for each component. Thus, it can determine how and which parts of the embedding to differentiate for the two directions.

Next, in the weighted version of LAS-GNN, we incorporate weights in this approach by replacing $\mathit{Linear}(1)$ in Equation 2 by the weight $\mathit{Linear}(w(u, v))$ for v's incoming neighbours u, and $\mathit{Linear}(-1)$ by $\mathit{Linear}(-w(v, u))$ for v's outgoing neighbours u. As a result, the message transferred over that edge is multiplied by the linear representation of the corresponding weight. This way, the weight determines the importance of the message to its recipient.

We argue that the integration of edge weights and signed message passing is particularly intuitive in financial networks, where negative weights correspond to negative transactions in the opposite direction. This edge weight multiplication is an alternative to

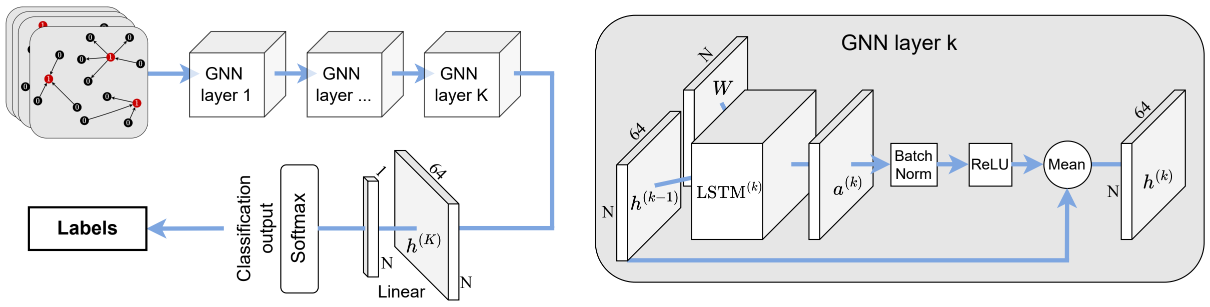


Figure 4: The architecture of LAS-GNN. Left: The global architecture. Right: A single GNN layer, shown with LSTM aggregation.

more standard approaches that treat the weight as an edge feature. It can also serve as a domain expert-defined attention coefficient [44].

Finally, we use *Long Short-Term Memory* (LSTM) [20] to capture temporal information. LSTM was proposed by Hamilton et al. [18] as a possible AGGREGATE function, but applied to a random permutation of the node's neighbours. Instead, we aim to capture the network dynamics and aggregate messages from neighbours in increasing order of timestamp corresponding to the edge. More formally, we define the ordered sequence of neighbours of node v as $N^T(v) = \langle u_1, \ldots, u_n \rangle$, where $u_i \in N(v)$ is the ith neighbour of v in the sequence, such that $\text{time}(u_i, v) < \text{time}(u_j, v)$ for all $i < j$. Then, $\boldsymbol{a}_v^{(k)} = \text{LSTM}^{(k)}\left(\langle \boldsymbol{h}_u^{(k-1)} \mid u \in N^T(v)\rangle\right)$. Note that this ordering is agnostic to edge direction.

We employ one LSTM for each layer k. This allows the model to learn from different sequential orderings at each depth. The LSTMs we employ consist of a single layer of the same size as the GNN layers. The weights of the LSTMs are updated simultaneously with the other weights of the GNN during backpropagation.

We refer to our ultimate GNN model, which includes ego IDs, signed message passing, edge weight multiplication, and LSTM aggregation, as **L**STM-based **A**ggregation and **S**igned message passing GNN (LAS-GNN) (see Figure 4). Formally,

$$\begin{aligned} \boldsymbol{a}_v^{(k)} &= \text{LSTM}^{(k)}\left(\langle \boldsymbol{d}_{u,v} \odot \boldsymbol{h}_u^{(k-1)} \mid u \in N^T(v)\rangle\right) \\ \boldsymbol{h}_v^{(k)} &= \textsc{Mean}\left(\boldsymbol{h}_v^{(k-1)}, \quad \textsc{ReLU}(\textsc{Norm}^{(k)}(\boldsymbol{a}_v^{(k)}))\right). \end{aligned} \tag{3}$$

We mention some implementation details. Note that ego IDs fail when node embeddings are calculated for all nodes simultaneously, since the GNN cannot differentiate between nodes that all have ego ID 1. By conducting *mini-batch training*, we only consider the embedding generation for a small group of seed nodes (i.e. a batch) at a time. For each batch, a small number of nodes are randomly sampled, their neighbourhoods are extracted in a breadth-first manner, and message passing is conducted in the corresponding subgraphs. If the sampled neighbourhoods of these seed nodes do not overlap, this issue with ego IDs does not arise. Thus, we ensure disjoint sampling of neighbourhoods, leading to disjoint subgraphs.

After K rounds of message passing, the final node embeddings are transformed by a linear layer, and then normalized by a softmax activation function to obtain the classification probabilities.

Regarding training, we apply the Adam optimizer [25] along with a standard binary cross-entropy loss function. To address data imbalance, we used the ratio of positive class labels out of the total number of samples as a weight in the loss calculation.

4 Experimental Setup

In our experiments, we focus on measuring the effectivity of the three new ingredients to our GNN framework: signed message passing, weighted message passing, and LSTM aggregation. We apply our method on simple directed graphs (we do allow bidirected edges) in temporal and weighted settings. In our networks (described below), we aim to find several of the motifs of Suzumura and Kanezashi [43]. We focus on the gather-scatter (GS), scatter-gather (SG), and directed cycle motifs in the temporal setting; their key-nodes are marked in red in Figure 3. We denote by Cℓ the directed cycle with ℓ vertices. Initially, we only consider C3. In the cycle, we mark an arbitrary node of the cycle as key-node (this is our minimal feedback node set). In our preliminary experiments in Section 5, we also consider fan-in (FI), fan-out (FO), and bipartite clique (BC). We call all these motifs our *target motifs*.

4.1 Datasets

For the benefit of a controlled experiment, we work with artificial, generated network. Specifically, we create two datasets as follows.

WSM Dataset. We generate artificial networks using the *Watts-Strogatz model* (WSM) [45]. This yields graphs with the target motifs occurring naturally, rather than being injected into the network post hoc, which risks skewing the random distribution and creating bias that affects detection algorithms. Since WSM is undirected by default, we enhance it to create directed graphs by directing edges randomly. Our WSM then has four parameters: total number of nodes n, average degree k, probability of rewiring an edge p_{rewire}, and probability of edge reciprocation p_{recip} (thus $(1 - p_{\text{recip}})/2$ are the probabilities of each direction separately). After preliminary experiments to check motif occurrence, we set $n = 10\,000$, $k = 6$, $p_{\text{rewire}} = 0.4$, $p_{\text{recip}} = 0.1$. We generate separate WSM networks for training, validation, and testing (all with the same parameter values). This ensures there is no information leakage while testing our inductive approach.

LFR Dataset. We generate artificial networks using the *LFR model* [29]. This, in contrast to WSM graphs [3], yields graphs with a power-law degree distribution and communities [29]. It has the following parameters: total number of nodes n, power-law exponent of the degree distribution γ, power-law exponent of the community size distribution β, mixing parameter μ (the fraction of edges a node shares with nodes outside its community), average degree k, minimum degree $k_{\min}$ and maximum degree $k_{\max}$, and minimum community size $s_{\min}$ and maximum community size $s_{\max}$. We set $n = 100\,000$, $\gamma = 3, \beta = 1.5, \mu = 0.1, k = 6$ (same as for WSM), $k_{\min} = 0, k_{\max} = 20$ (this yielded networks with degree 30), $s_{\min} = 5$, $s_{\max} = 500$. We generate separate LFR networks for training, validation, and testing (all with the same parameter values).

For both WSM and LFR data, we generate times by assigning each edge a unique timestamp using a random permutation of $\{1, \ldots, m\}$, where m is the number of edges in the graph. For the WSM data, we also generate the weight of each edge uniformly at random in $[0, 1]$. To ensure that sufficiently many target motifs occur as a weighted motif, we sample half of the naturally occurring motifs and assign weights uniformly at random in $[r, 1]$, where r is the threshold of weighted motif occurrence. Note that further occurrences may appear by chance.

For the ground truth, we employ an exhaustive search and classify the key-node of a (weighted and/or temporal) target motif as suspicious (1), while all other nodes are marked non-suspicious (0).

4.2 Compared Models

In order to understand the impact of each component that makes up LAS-GNN, we implement and test several versions of the framework. We compare them with a similarly diverse set of models based on a GIN architecture as proposed by Egressy et. al [12].

Baseline. For the unweighted, non-temporal setting, we implement a GNN model with ADD aggregator and vary the directed message passing mechanism and turn on/off the ego ID. These configurations are used in our preliminary experiments (Section 5).

LAS-GNN. For the LAS-GNN algorithm, after a grid search on validation data, we settled on a learning rate of 0.001, an embedding size of 64, and a batch size of 32.

In preliminary experiments, we found that four GNN layers suffice to detect all target motifs. This also appears intuitively to be the minimum number of layers required. For instance, in a C4 motif, the ego ID must traverse four edges to return to its original seed node. We also found that using a default element-wise sum aggregator can be advantageous in the first GNN layer. Hence, in our experiments, we use this default aggregator as the first layer, followed by three LSTM layers.

We consider three variants of LAS-GNN. All use LSTM aggregation: *LAS-GNN* is our ultimate model, bringing together signed message passing and temporally ordered LSTM aggregation. *LAS-GNN-Random* uses a LSTM aggregator but orders the neighbours randomly instead of temporally, and uses signed message passing *LAS-GNN-HMP* applies heterogeneous message passing (instead of signed message passing) and applies temporally sorted LSTM aggregation to incoming and outgoing neighbours separately. All LAS-GNN models are used with the same set of hyperparameters as specified above.

GIN+Ego+HMP. As state-of-the-art, we compare our approach to the model of Egressy et al. [11], stripped of the port-numbering component, since we work with simple graphs. Specifically, the model is a GIN architecture with edge features [21], ego IDs and heterogeneous message passing, and we call it *GIN+Ego+HMP*. The ability of the model to represent edge features is important as this provides us with the means of seamlessly inputting the edge weights and timestamps as edge features. We refer to the weighted, temporal model as *GIN+Ego+HMP*^{wt}. The timestamp feature is L_2 normalized since using absolute timestamp values results in worse performance. The MLP used is a two-layer neural network with 64 nodes in each layer. The ϵ-parameter is set to 0.

Model	ego ID	MP	Weight	Time	Agg
GNN	✓	*any*	✗	✗	ADD
GIN+Ego+HMP	✓	hetero	✗	✗	ADD
GIN+Ego+HMP^{wt}	✓	hetero	feature	feature	ADD
LAS-GNN	✓	signed	mult.	sorted	LSTM
- Random	✓	signed	✗	✗	LSTM
- HMP	✓	hetero	✗	sorted	LSTM

Table 1: Overview of the models used in our experiments.

We refer to Table 1 for an overview of all compared models.

For implementation we use `PyTorch Geometric` [15]. All experiments were run on a server running Ubuntu 22.04 LTS with Intel Xeon Gold 6238R CPU @ 2.20GHz, Nvidia A100 PCIe 40 GB GPU, and 256GB of RAM. We run each experiment for 100 epochs. We assume convergence and apply early stopping once the validation loss does not improve for 20 epochs. All reported results are averaged over five networks; each has a different seed.

5 Preliminary Experiments

Before evaluating LAS-GNN on temporal, weighted networks, we report on our experiments conducted to confirm that ego IDs, signed-message passing, and the LSTM aggregator are working in line with state-of-the-art on unweighted, non-temporal networks. We perform these experiments on our WSM graphs.

We first isolate the effectivity of the employed message passing approach and ego IDs (see Table 2). These experiments partially reproduce results of Egressy et al. [11], confirming that heterogeneous message passing combined with ego IDs to the node features (GNN-Hetero+ego ID) is essential to correctly identify the target motifs, offering a substantial improvement over bidirectional and directional message passing. The poor performance of bidirectional message-passing is explained by the fact that it does not account for directionality, while all the target motifs are directed. Importantly, we observe that our alternative approach of signed message passing seems to be very effective. Echoing the results for heterogeneous message passing, it achieves perfect detection results in directed graphs with reciprocating edges.

Next, to ensure that using an LSTM does not harm detection results, we compare GIN+Ego+HMP to LAS-GNN-Random. From the perfect results in Table 2, we conclude that using LSTM aggregators instead of MLPs does not deteriorate motif detection capabilities, even when learning a sequential order is not required.

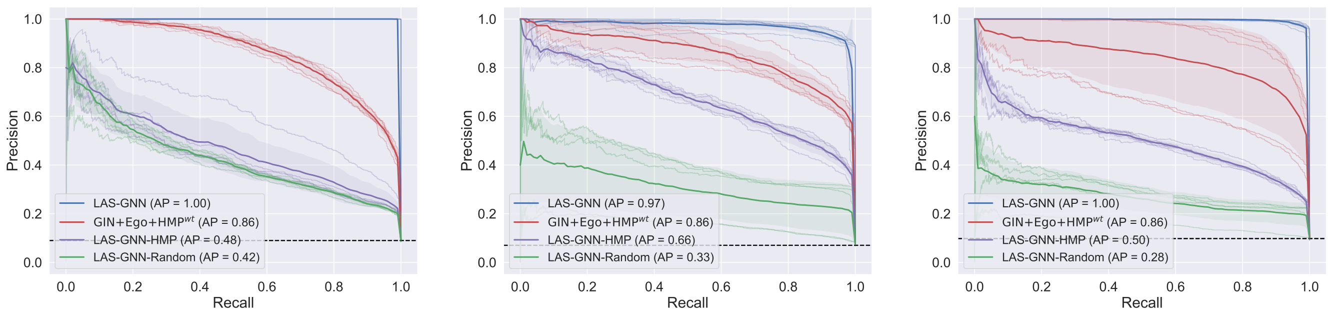


Figure 5: PR curves for temporal motif detection on WSM graphs with 10K nodes. The filled area marks the standard deviation. *Left:* GS motif, imb.$\approx$ 9.0%. *Middle:* SG motif, imb.$\approx$ 7.0%. *Right:* C3 motif, imb.$\approx$ 9.8%.

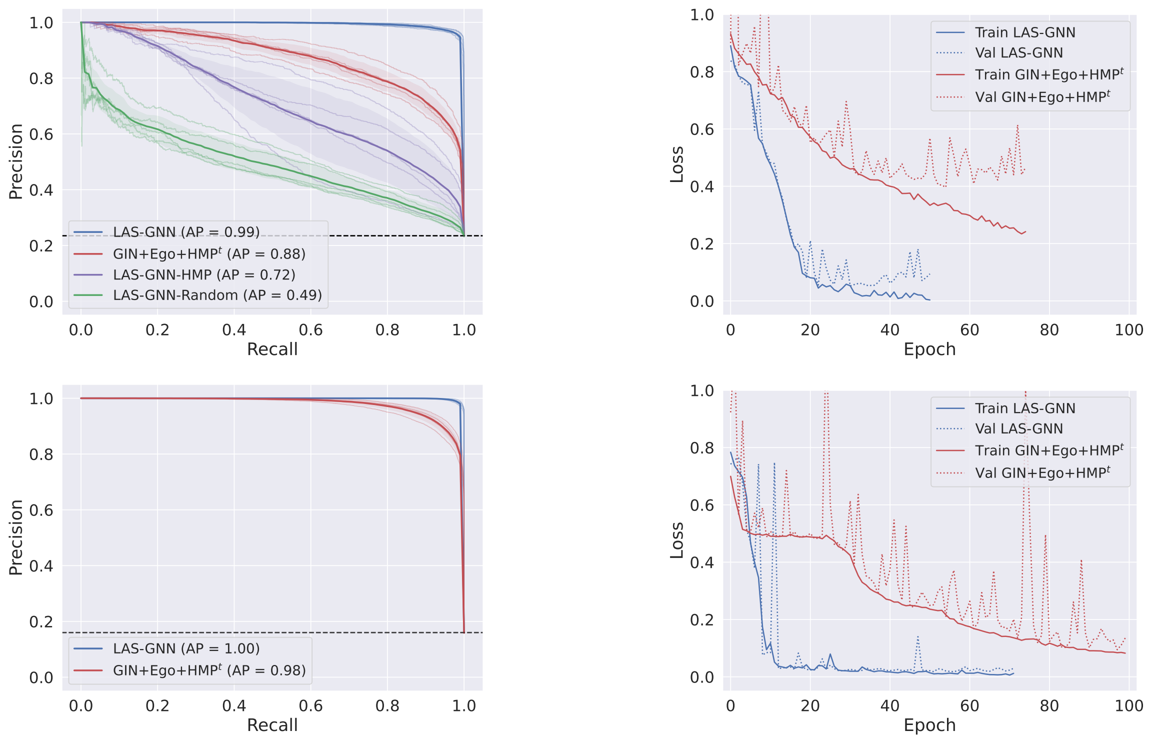


Figure 6: *Top Left:* PR curves for unweighted temporal motif detection, simultaneously for GS, SG, and C3 motifs, on WSM graphs with 10 000 nodes. Imb.: $\approx$ 23.5%. The filled area marks the standard deviation. *Top right:* Training and validation loss of a single run in this setting. *Bottom left:* PR curves for unweighted temporal motif detection, simultaneously for GS, SG, and C3 motifs, on LFR graphs with 100 000 nodes. Imb.: $\approx$ 16.0%. *Bottom right:* Training and validation loss of a single run in this setting.

6 Results

In this section, we evaluate LAS-GNN on the weighted and temporal motif detection tasks that it has been designed for. Our focus is on the motifs illustrated in Figure 3, specifically: GS, SG, and C3.

Unweighted, Temporal. Figure 5 shows the results from our experiments on WSM graphs when combining the LSTM aggregator and signed message passing, as formalized in Equation 3. LAS-GNN can almost perfectly detect these temporal motifs.

The failure of LAS-GNN-Random is inevitable: it does not use temporal information. Still, this shows that LSTM aggregation only works when the edge ordering is respected. LAS-GNN-HMP does make use of temporal information, but its lacking performance shows that its separate aggregation is incompatible with learning the sequential ordering of temporal motifs. GIN+Egp+HMP^{wt} manages to learn from the temporal motif examples provided during training where timestamps are provided as edge features. However,

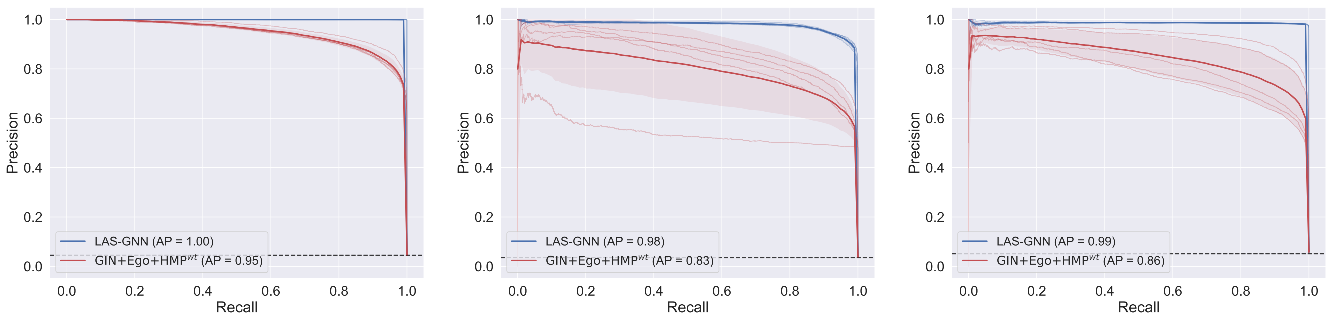


Figure 7: PR curves for weighted temporal motif detection on WSM graphs with 100 000 nodes. The filled area marks the standard deviation. *Left:* GS motif, imb.≈ 4.5%. *Middle:* SG motif, imb.≈ 3.5%. *Right:* C3 motif, imb.≈ 5.1%.

Model	FI	FO	GS	SG	BC	C3	C4
GNN-Bidirect.	0.64	0.67	0.73	0.48	0.47	0.71	0.64
+ego ID	0.64	0.65	0.73	0.49	0.49	0.73	0.68
GNN-Direct.	**1.0**	0.63	0.78	0.57	0.55	0.67	0.61
+ego ID	**1.0**	0.68	0.80	0.62	0.58	**1.0**	0.75
GNN-Hetero.	**1.0**	**1.0**	**1.0**	0.63	0.64	0.85	0.70
+ego ID	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
GNN-Signed	**1.0**	**1.0**	**1.0**	0.64	0.63	0.86	0.71
+ego ID	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
GIN+Ego+HMP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
LAS-GNN-Random	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Imb. (%)	*42.5*	*42.5*	*45.2*	*23.8*	*23.7*	*45.7*	*40.7*

Table 2: Average $F1$ scores for the unweighted, non-temporal setting, on WSM graphs with 10 000 nodes, after 10 epochs. Standard deviations are all $\leq$ 0.03. Perfect detection results are in bold.

Model	C3	C4	C5	C6
GIN+Ego+HMP	0.93 ± .006	0.81 ± .126	0.73 ± .094	0.49 ± .055
LAS-GNN	0.99 ± .002	0.98 ± .001	0.95 ± .004	0.87 ± .010

Table 3: F1 scores for temporal motif detection of longer cycles on WSM graphs. This task is balanced: we sampled 2000 temporal and 2000 non-temporal cycle motifs at random.

besides achieving a lower average precision (0.86), the PR curve shows that performance can be inconsistent across multiple runs.

It is important to observe that our GNN architecture is not specialized to detect a specific motif. To investigate this, we also consider networks where we mix the GS, SG, and C3 motifs. That is, we label all key nodes in any of these three motifs by 1. Figure 6 (top left) indeed shows that the LAS-GNN model achieves near perfect performance in this complex setting and can handle multiple motifs at the same time, in contrast to the other models. The comparison to LAS-GNN-HMP and LAS-GNN-Random confirms that it is the combination of signed message passing and LSTM in the Aggregator function of LAS-GNN that accounts for its performance. As alluded to earlier, Figure 6 (top right) shows that the training and validation loss of the GIN+Ego+HMP^{wt} model suffers considerable instability and it takes considerable longer for it to converge.

We also consider the performance when detecting longer cycles (up to length 6). We use K layers for cycles of length K. We use much larger networks: up to 4M nodes for C6. This leads to a very large imbalance; therefore, we report on the *balanced* task, where we sampled 2000 suspicious and 2000 non-suspicious nodes from the graph at random. Table 3 shows that the average F1 score of both models decreases as the cycle length increases. It seems that having the timestamps as edge features is much less robust when detecting longer cycles than utilizing LSTM aggregation as done in LAS-GNN. This could mean that the latter is a more promising approach for larger subgraph motifs that require message passing at further depth.

Weighted, Temporal. Finally, we add weights to our WSM graphs. We experimented with various weight threshold r values, and the performance sees a relative decay as r increases. In the interest of brevity, we report results obtained with a weight threshold $r = 0.8$. Recall that we suspiciously weigh around half of the occurring motifs. To ensure a fair task, we set the weights of an equal number of non-temporally ordered motifs to be suspicious. We train, validate, and test on networks with 100K nodes, using a batch size of 512, to accommodate the more severe class imbalance of this task.

Figure 7 shows our results for the GS, SG, and C3 motifs. For the GS motif, both models perform almost equally well. Note that when we inspect the training and validation loss progression of a single run, GIN+Ego+HMP^{wt} needs more training rounds: 40 versus just 15 epochs for LAS-GNN. Also, the validation loss of the GIN+Ego+HMP^{wt} model exhibits considerable instability from that point onward. The F1-score is 1.000 for LAS-GNN and 0.908 for GIN+Ego+HMP^{wt}. Similar behavior occurs for the C3 motif. The SG motif, however, seems more difficult to detect for both models. Still, the F1-score is 0.934 for LAS-GNN and 0.839 for GIN+Ego+HMP^{wt}.

7 Discussion

Compared to the GIN+Ego+HMP^{wt} model of Egressy et al. [11], the LAS-GNN model offers clear advantages in temporal settings.

Our results suggest that, even though all temporal information is available through the edge features, GIN+Ego+HMP^{wt} struggles to learn from this information effectively. Note that GIN+Ego+HMP^{wt} must independently determine that the timestamps represent a sequential order, whereas LAS-GNN explicitly models this in its aggregation mechanism. LAS-GNN also seems more robust than GIN+Ego+HMP^{wt} when detecting larger cyclic motifs.

It may also be interesting to compare LAS-GNN against further GNNs, such as done in the extensive benchmarking by Egressy et al. [11]. However, the GIN+Ego+HMP^{wt} model of Egressy et al. [11] was shown to significantly outperform other available models in the non-temporal setting, making this the most relevant baseline. Similarly, while a comparison to temporal GNNs could be interesting, existing temporal GNNs do not fit our chronological setting, as far as we are aware, as already discussed in the introduction.

The experiments we performed may not yet showcase the full potential of LAS-GNN. A limitation is that our experiments focused on simple directed graphs (although we allow bidirectional edges). Using port numbering [40], we may be able to extend our model to directed multigraphs and demonstrate the potential of LAS-GNN on the extended testbed of Egressy et al. [11]. In particular, they used a variant of the Watts-Strogatz model with parallel edges and directed multigraphs derived from the AMLworld simulator [1]. In future work, we aim to test LAS-GNN further on data sets based on AMLSim [43], AMLworld [1], and Elliptic [47].

In the same vein, a limitation of our use of the Watts-Strogatz model is that it produces a Poisson degree distribution [3], while real networks are often scale-free networks with a power-law degree distribution [2], enabling the presence of *hubs* – nodes with high degree. In real financial networks, hubs often correspond to non-suspicious companies [42], so this may not be a substantial issue. Still, we did perform some initial experiments on our LFR dataset. As shown in Figure 6 (bottom), LAS-GNN detects motifs very well for mild power-law exponents and small maximum degree. While our LFR dataset uses $\gamma = 3$, we noticed the same excellent results down to $\gamma = 2.5$, but ran out of memory for lower values of γ. Moreover, performance seems to degrade as the exponent of the power-law decreases and hubs become more pronounced. When increasing the maximum degree to 100 (keeping $\gamma = 3$), using four layers required too much memory. To detect C3, we require only three layers, but notice performance issues already in the non-temporal setting: while ADD aggregation works well, LSTM aggregation performs poorly and the model does not seem to learn. There is some evidence in the literature that all GNNs suffer from similar performance issues, although this can sometimes be mitigated [35, 56]. Moreover, we seem to run into issues with LSTM aggregation specifically. We hope to extend LAS-GNN to still perform well in these situations.

8 Conclusion

We summarize our work: (i) we provided a new definition of temporal graph motifs that aligns with the flow of funds in money laundering; (ii) we proposed a directed GNN message passing mechanism that achieves state-of-the-art performance while being compatible with edge timestamps; (iii) we proposed a new GNN model for temporal motif finding that shows consistent superiority to straightforward adaptations of static state-of-the-art motif finding solutions to temporal graphs; (iv) our model is to the best of our knowledge the first to leverage the permutation sensitivity of an LSTM aggregator in GNNs to capture temporal interaction of nodes.

References

[1] Erik Altman, Jovan Blanuša, Luc Von Niederhäusern, Béni Egressy, Andreea Anghel, and Kubilay Atasu. 2023. Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. In *Proc. NeurIPS 2023*.

[2] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286 (1999), 509–512. Issue 5439.

[3] Alain Barrat and Martin Weigt. 2000. On the properties of small-world network models. *Eur. Phys. J. B* 13 (2000), 547–560.

[4] Jovan Blanusa, Maximo Cravero Baraja, Andreea Anghel, Luc von Niederhäusern, Erik R. Altman, Haris Pozidis, and Kubilay Atasu. 2024. Graph Feature Preprocessor: Real-time Subgraph-based Feature Extraction for Financial Crime Detection. In *Proc. ICAIF 2024*. ACM, 222–230.

[5] Jovan Blanusa, Paolo Ienne, and Kubilay Atasu. 2022. Scalable Fine-Grained Parallel Cycle Enumeration Algorithms. In *Proc. SPAA '22*. ACM, 247–258.

[6] Kevin Buehler. 2019. Transforming approaches to AML and financial crime. McKinsey.

[7] Z. Chai, Y. Yang, J. Dan, S. Tian, C. Meng, W. Wang, and Y. Sun. 2023. Towards Learning to Discover Money Laundering Sub-network in Massive Transaction Network. In *Proc. AAAI 2023*. AAAI Press, 14153–14160.

[8] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. 2020. Can Graph Neural Networks Count Substructures?. In *Proc. NeurIPS 2020*.

[9] Zhiyuan Chen, Le Dinh Van Khoa, Ee Na Teoh, Amril Nazir, Ettikan Kandasamy Karuppiah, and Kim Sim Lam. 2018. Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Knowl. Inf. Syst.* 57, 2 (2018), 245–285.

[10] Dawei Cheng, Yujia Ye, Sheng Xiang, Zhenwei Ma, Ying Zhang, and Changjun Jiang. 2023. Anti-Money Laundering by Group-Aware Deep Graph Learning. *IEEE Trans. Knowl. Data Eng.* 35, 12 (2023), 12444–12457.

[11] Béni Egressy, Luc Von Niederhäusern, Jovan Blanuša, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. 2024. Provably Powerful Graph Neural Networks for Directed Multigraphs. In *Proc. AAAI 2024*. AAAI Press, 11838–11846.

[12] Béni Egressy and Roger Wattenhofer. 2022. Graph neural networks with precomputed node features. *arXiv:2206.00637* (2022).

[13] European Parliament and Council of European Union. 2024. Regulation (EU) 2024/1624. http://data.europa.eu/eli/reg/2024/1624/oj

[14] ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, and Shuibing He. 2024. A Comprehensive Survey of Dynamic Graph Neural Networks: Models, Frameworks, Benchmarks, Experiments and Challenges. *CoRR* abs/2405.00476 (2024).

[15] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv:1903.02428* (2019).

[16] Oscar M Granados and Andrés Vargas. 2022. The geometry of suspicious money laundering activities in financial networks. *EPJ Data Science* 11, 1 (2022), 6.

[17] László Hajdu and Miklós Krész. 2020. Temporal Network Analytics for Fraud Detection in the Banking Sector. In *Proc. ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium (CCIS, Vol. 1260)*. Springer, 145–157.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proc. NeurIPS 2017*. 1024–1034.

[19] Jing He, Jiao Tian, Yuanyuan Wu, Xinyi Cia, Kai Zhang, Mengjiao Guo, Hui Zheng, Junfeng Wu, and Yimu Ji. 2021. An efficient solution to detect common topologies in money launderings based on coupling and connection. *IEEE Intell. Syst.* 36, 1 (2021), 64–74.

[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *Proc. ICLR 2020*. OpenReview.net.

[22] Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. 2022. Going deeper into permutation-sensitive graph neural networks. In *Proc. ICML 2022 (PMLR, Vol. 162)*. 9377–9409.

[23] Guillaume Jaume, An-phi Nguyen, María Rodríguez Martínez, Jean-Philippe Thiran, and Maria Gabrani. 2019. edGNN: a Simple and Powerful GNN for Directed Labeled Graphs. *CoRR* abs/1904.08745 (2019).

[24] Ali Jazayeri and Christopher C. Yang. 2020. Motif discovery algorithms in static and temporal networks: A survey. *J. Complex Networks* 8, 4 (2020).

[25] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR 2015*.

[26] Eren Kurshan, Hongda Shen, and Haojie Yu. 2020. Financial Crime & Fraud Detection Using Graph Computing: Application Considerations & Outlook. In *Proc. TransAI 2020*. IEEE, 125–130.

[27] Dattatray Vishnu Kute, Biswajeet Pradhan, Nagesh Shukla, and Abdullah M. Alamri. 2021. Deep Learning and Explainable Artificial Intelligence Techniques

Applied for Detecting Money Laundering-A Critical Review. *IEEE Access* 9 (2021), 82300–82317.

[28] Nevine Makram Labib, Mohammed Abo Rizka, and Amr Ehab Muhammed Shokry. 2020. Survey of machine learning approaches of anti-money laundering techniques to counter terrorism finance. In *Proc. ITAF 2019 (LNNS, Vol. 114)*. 73–87.

[29] Andreas Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* 78 (2008), 046110.

[30] Jiantao Li, Jianpeng Qi, Yueling Huang, Lei Cao, Yanwei Yu, and Junyu Dong. 2024. MoTTo: Scalable Motif Counting with Time-aware Topology Constraint for Large-scale Temporal Graphs. In *Proc. CIKM 2024*. ACM, 1195–1204.

[31] Xujia Li, Yuan Li, Xueying Mo, Hebing Xiao, Yanyan Shen, and Lei Chen. 2023. Diga: Guided diffusion model for graph recovery in anti-money laundering. In *Proc. SIGKDD 2023*. ACM, 4404–4413.

[32] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. 2020. FlowScope: Spotting Money Laundering Based on Graphs. In *Proc. AAAI 2020*. AAAI Press, 4731–4738.

[33] Yuchen Li, Zhengzhi Lou, Yu Shi, and Jiawei Han. 2018. Temporal Motifs in Heterogeneous Information Networks. In *Proc. MLG 2018*.

[34] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. 2023. Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities. *Trans. Mach. Learn. Res.* (2023).

[35] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2022. Is Homophily a Necessity for Graph Neural Networks?. In *Proc. ICLR 2022*. OpenReview.net.

[36] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *Proc. WSDM 2017*. ACM, 601–610.

[37] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv:2006.10637* (2020).

[38] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. 2023. Edge directionality improves learning on heterophilic graphs. In *Proc. LoG 2023 (PMLR, Vol. 231)*. 25.

[39] Ahmet Erdem Sarıyüce. 2025. A powerful lens for temporal network analysis: temporal motifs. *Discover Data* 3 (2025), 14.

[40] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. 2019. Approximation ratios of graph neural networks for combinatorial problems. *Proc. NeurIPS 2019*, 4083–4092.

[41] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2021. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access* 9 (2021), 79143–79168.

[42] Michele Starnini, Charalampos E Tsourakakis, Maryam Zamanipour, André Panisson, Walter Allasia, Marco Fornasiero, Laura Li Puma, Valeria Ricci, Silvia Ronchiadin, Angela Ugrinoska, et al. 2021. Smurf-based anti-money laundering in time-evolving transaction networks. In *Proc. ECML PKDD 2021 (LNCS, Vol. 12978)*. Springer, 171–186.

[43] Toyotaro Suzumura and Hiroki Kanezashi. 2021. Anti-Money Laundering Datasets: InPlusLab Anti-Money Laundering DataDatasets. http://github.com/IBM/AMLSim/.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proc. ICLR 2018*. OpenReview.net.

[45] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *Nature* 393, 6684 (1998), 440–442.

[46] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. 2018. Scalable graph learning for anti-money laundering: A first look. *arXiv:1812.00076* (2018).

[47] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv:1908.02591* (2019).

[48] Zhihao Wen and Yuan Fang. 2022. TREND: TempoRal Event and Node Dynamics for Graph Representation Learning. In *Proc. WWW 2022*. ACM, 1159–1169.

[49] Jiajing Wu, Jieli Liu, Weili Chen, Huawei Huang, Zibin Zheng, and Yan Zhang. 2022. Detecting Mixing Services via Mining Bitcoin Transaction Network With Hybrid Motifs. *IEEE Trans. Syst. Man Cybern. Syst.* 52, 4 (2022), 2237–2249.

[50] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24.

[51] Leshanshui Yang, Clément Chatelain, and Sébastien Adam. 2024. Dynamic Graph Representation Learning With Neural Networks: A Survey. *IEEE Access* 12 (2024), 43460–43484.

[52] Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. 2020. Neural Subgraph Matching. *arXiv:2007.03092* (2020).

[53] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. 2021. Identity-aware graph neural networks. In *Proc. AAAI 2021*, Vol. 35. 10737–10745.

[54] Yichao Yuan, Haojie Ye, Sanketh Vedula, Wynn Kaza, and Nishil Talati. 2023. Everest: GPU-Accelerated System For Mining Temporal Motifs. *Proc. VLDB Endow.* 17, 2 (2023), 162–174.

[55] Yanping Zheng, Lu Yi, and Zhewei Wei. 2025. A survey of dynamic graph neural networks. *Frontiers of Computer Science* 19, 6 (2025), 196323.

[56] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *Proc. NeurIPS 2020*.