IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 February 2024, accepted 19 March 2024, date of publication 1 April 2024, date of current version 12 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3383784

RESEARCH ARTICLE

Scalable Semi-Supervised Graph Learning Techniques for Anti Money Laundering

MD. REZAUL KARIM[1,2], FELIX HERMSEN[1,2], SISAY ADUGNA CHALA[1], PAOLA DE PERTHUIS[3,4], AND AVIKARSHA MANDAL[2]

[1]Information Systems and Databases, RWTH Aachen University, 52074 Aachen, Germany
[2]Department of Data Science and Artificial Intelligence, Fraunhofer FIT, 53757 Sankt Augustin, Germany
[3]École Normale Supérieure (ENS), 75005 Paris, France
[4]Cosmian, 75008 Paris, France

Corresponding author: Md. Rezaul Karim (rezaul.karim@rwth-aachen.de)

This work was supported by the Agence Nationale de la Recherche (ANR) and the Bundesministerium für Bildung und Forschung (BMBF) under the Franco-German AI Call for the project "CRYPTO4GRAPH-AI", grant number 01IS21100A.

ABSTRACT Money laundering is the process by which criminals move large sums of illicit money to hidden locations and integrate them as legal funds through existing financial services. The United Nations (UN) estimates that 2 to 5% of global GDP, which is approximately 0.8 to 2.0 trillion dollars, is laundered globally every year. Therefore, accurately identifying such globally alarming activities is crucial for enforcing anti-money laundering (AML) measures. Numerous techniques have been proposed to detect money laundering from transaction graphs of money transfers between bank accounts by analysing the structural and behavioural dynamics of their corresponding dense subgraphs. However, these techniques often do not consider that money laundering usually involves high-volume flows of funds through chains of bank accounts. Moreover, most AML approaches either result in lower detection accuracy or incur higher computational costs, making them less reliable and suitable for real financial systems. Consequently, only a fraction of money laundering activities can be detected and prevented. In this paper, we propose an efficient approach to AML by employing semi-supervised graph learning techniques on a large-scale financial transactional graph in both pipeline settings (i.e., graph embedding models are first trained to generate node embeddings that are combined with additional topological graph features to train binary classifiers) and end-to-end settings (i.e., node classification is performed by training SkipGCN, FastGCN, and EvolveGCN without requiring separate classifiers) to identify nodes involved in potential money laundering activities. We evaluate our approach on four datasets: *AMLSim*, *Elliptic*, *IBM AML*, and *SynthAML*, with a view to scalability and practicality for real financial systems. Further, we provide local (e.g., how money is laundered between nodes) and global (e.g., what factors contribute to money laundering) explanations of the predictions by highlighting the predominant factors in money laundering cases and elucidating the mechanisms of illicit fund transfers between nodes to enhance the interpretability and transparency of the AML models. Experimental results suggest that our approach is scalable and effective at detecting money laundering from real and synthetic transaction graphs.

INDEX TERMS Anti-money laundering, machine learning on graphs, graph embedding.

I. INTRODUCTION

Money laundering is a globally challenging economic concern. The *UN Vienna 1988 Convention* describes it as *"the conversion or transfer of property, knowing that such property is derived from any offence (s), to conceal or disguise the illicit origin of the property or of assisting any person who is involved in such offence (s) to evade the legal consequences of his actions."*. Being a global issue, money laundering results in approximately 0.8 to 2.0$ trillion being laundered every year, which equates to 2 to 5% of the world's GDP [1], [2]. Being a global issue, money laundering results in approximately 0.8 to 2.0$ trillion being laundered every year, which equates to 2 to 5% of the world's GDP [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Ines Domingues.

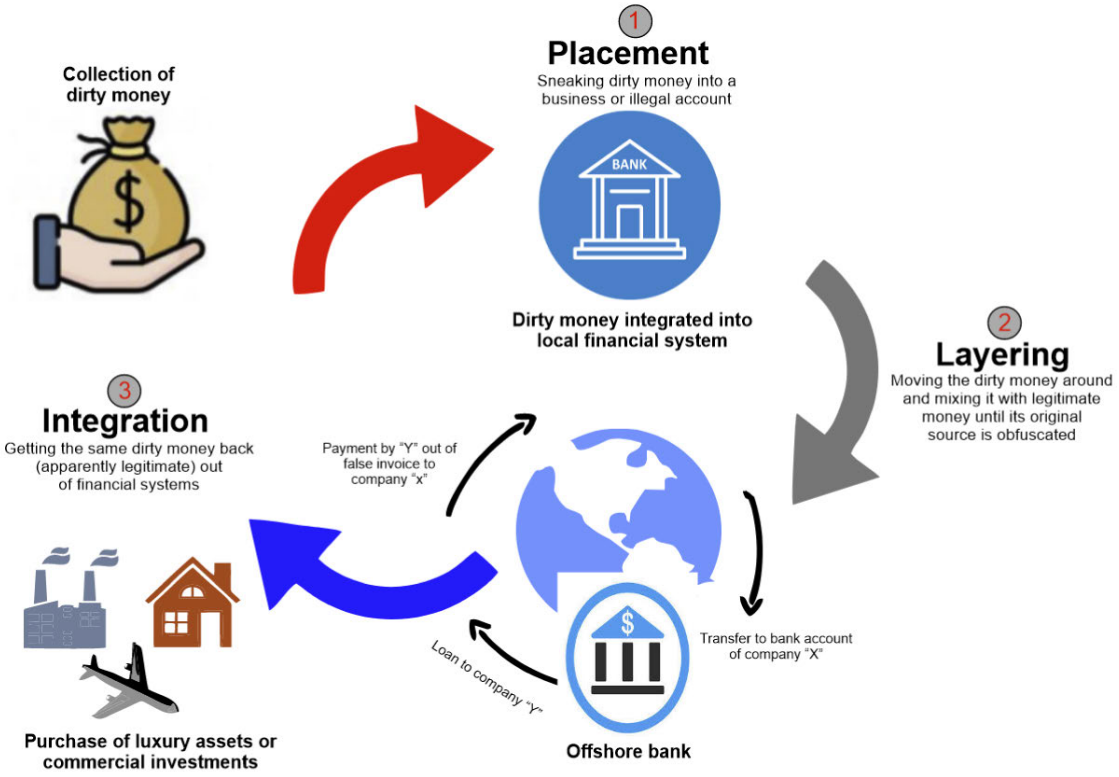


FIGURE 1. Main stages in money laundering activities (source: recreated based on [3]).

As shown in fig. 1, money laundering involves three main stages: *placement*, *layering*, and *integration*. *Placement* is introducing illicit funds into financial systems. *Layering* is conducting complex transactions to obscure the origins. *Integration* is withdrawing the proceeds from a destination bank account and utilizing them for lawful purposes.

As shown in fig. 2, money laundering can be modelled by two major topologies: in the first topology, there is one source account, one destination account, and many middle accounts that form a vertical chain. First, the source splits the money among the middle accounts, which then sends the money to the destination account. In the second topology, there is one source account, one destination account, and a horizontal sequence of middle accounts. Each middle account transfers the whole amount of money to the next account in the sequence until it reaches the destination. Based on such topologies, a money transaction graph can be created by representing a single account as a vertex and a transaction between two accounts as an edge. Financial institutions are responsible for forensic analysis and complying with regulatory standards such as know your customer (KYC), transaction monitoring, suspending suspicious accounts, and submitting suspicious activities reports (SARs) to law enforcement agencies.[1]

Preventing criminals from moving such illicit funds through financial systems is known as anti-money laundering (AML). Since failure to AML compliance may impose severe penalties, financial organizations must identify illicit activities reliably. For this, rigorous analysis is carried out using sophisticated techniques to determine whether a SAR needs to be filed or whether the account in question should be suspended. However, the current AML approaches face several limitations and challenges. One of the main challenges is the lack of access to real financial transaction data, which is highly restricted due to proprietary and privacy reasons [2]. First, access to real financial transaction data is highly restricted due to both proprietary and privacy reasons [2]. AML efforts suffer from a lack of real data (or at least, realistic synthetic data that mimics AML scenarios), labels as well as annotations of important attributes of real data [2]. This hinders the development and evaluation of AML methods, as they cannot be tested on realistic scenarios and data. The second challenge is the scarcity and the imbalance of labelled data, e.g., as the normal transactions and accounts vastly outnumber the illicit ones. A concrete example of two parties sharing the same IP address on a transaction graph does not necessarily mean they are physically connected since their IP addresses might be associated with a mobile network. However, highly skewed labelled data exhibit limited adaptability and unreliable results. The third challenge is that the ground truth of money laundering is hard to obtain and verify, leaving most real-life

[1]For example, the global framework for anti-money laundering (AML) is regulated by the Financial Action Task Force.

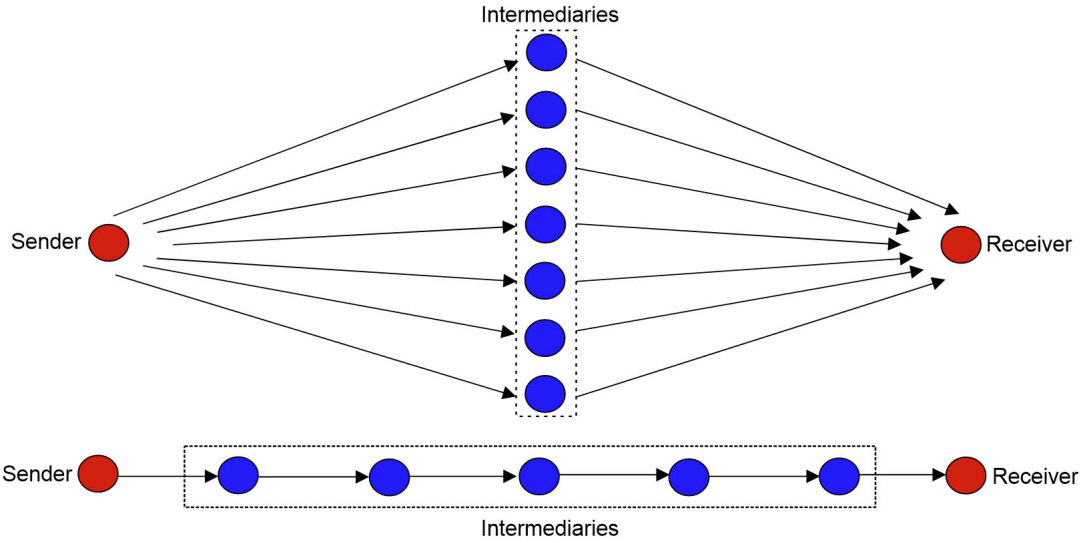


FIGURE 2. Two common topologies of money laundering networks (recreated based on literature [4]).

data noisy and unlabeled or sparsely labelled. However, the majority of the existing methods do money laundering detection in a supervised manner. This reduces the reliability and adaptability of the supervised learning methods, as they may suffer from overfitting or underfitting problems.

A fourth challenge is the complexity and the scalability of the graph model, which can contain billions of nodes and edges.[2] This requires efficient and scalable methods that can handle large-scale graphs and extract meaningful and discriminative features from them. The fifth challenge is that criminals often mask the true nature of their transactions using complicated account layering or multi-hop transactions, leaving the identification of money laundering a complex problem. Our empirical study outlines that most existing AML approaches yield either lower detection accuracy or are inscalable to large graphs, making them less reliable and impractical for real-life financial systems. Further, most of the existing methods are either too simple or too complex, resulting in lower detection accuracy or higher computational cost, making them less suitable and practical for real-life financial systems. As a consequence, the current AML approaches can only prevent and detect a tiny fraction of money laundering activities, leaving a huge gap between the actual and the reported cases.

On a potentially positive note datasets such as *Elliptic*, *IBM AML*, AMLSim [5], and AMLworld [2] have been produced that build multi-agent virtual worlds, where some of the agents are criminals with illicit income to launder. These are a great addition to tackle data scarcity issues and to advance AML research. Our empirical investigation suggests that one possible solution to deal with noisy data could be using the attention mechanism in the graph convolutional kernel and controlling the message passing based on the type of connection between these two parties and their (node) profiles. For labelled data scarcity, graph structural information can help significantly boost the performance of an AML model without much dependence on labels. On a similar note, Schlichtkrull et al. [6] show that node features are important, as domain knowledge of a node in a KG can be formulated as node features for the graph model, and rich node features are likely to boost the model performance. Our study found that node features could be important attributes and rich node features are likely to boost the model performance.

Inspired by the successes of recent graph-based approaches, the potential of topological node features, and the drawbacks of existing approaches, we employ semi-supervised learning techniques on large transaction graphs to detect money laundering. We hypothesize that similar to network analysis that involves predictions over edges, nodes in a transactional graph displaying distinct characteristics from regular nodes can be classified as potential money launders. Our semi-supervised learning approach involves using topological node features and annotations from alerts to embed graph nodes into a lower-dimensional vector space. The embeddings, along with other features are then used to train tree-based ensemble classifiers such as random forest (RF), extreme gradient boosted trees (XGBoost), and light gradient boosted machine (LightGBM) that predict the suspiciousness of a target node in potential money laundering activities based on its direct or indirect connections to nodes that are known to be suspicious. The overall contributions of this paper can be summarized as follows:

- We address a globally challenging economic concern - *money laundering* with a view to its criticality and importance towards deploying scalable and robust AML models into real financial systems.
- We employ state-of-the-art (SOTA) semi-supervised learning techniques on transaction graphs, where both spatial and temporal information, together with

[2]e.g., mappings of billions of edges between millions of entities.

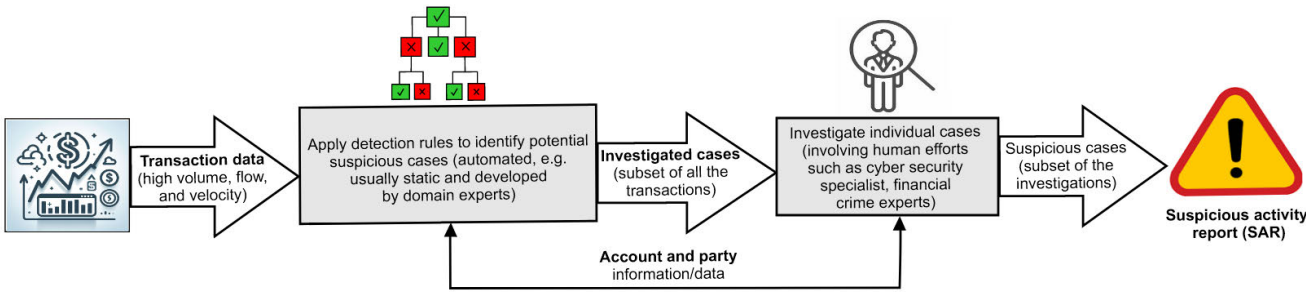


FIGURE 3. General workflow of rule-based suspicious activities reports (SARs) alerting.

topological graph node features, are modelled by combining the power of graph-based representation learning and tree-based ensemble models. Overall, our approach, to the best of our knowledge, is a "first-of-its-kind" approach in which both pipeline and end-to-end approaches are considered.

- Unlike existing approaches that evaluated their studies on a limited number of datasets, our approach considers evaluating the AML capability on several real and synthetic datasets, including *AMLSim*, *Elliptic*, *IBM AML*, and *SynthAML*, with a view to scalability and efficiency.
- We provide comprehensive evaluations of our approach, both quantitative and qualitatively.
- To improve the interpretability, we provide global (e.g., what factors contribute more in money laundering scenarios) and local (e.g., how money gets laundered between nodes) explanations.
- We provide guidelines on how an AML model can be deployed and integrated into real financial systems such as banks. Besides, we provide several outlooks for network security and financial crime analysts if they could employ semi-supervised graph learning on large-scale transaction graphs for effective identification of potential money laundering cases.
- We are in the process of making available Python notebooks and codes[3] that will help researchers reproduce the result interactively or extend the implementation by changing the network architectures or customising their datasets.

The rest of the paper is structured as follows: Section II critically reviews some related works. Section III describes our proposed approach in detail. Section IV reports some experiment results, including a comparative analysis with baseline models. Section V summarizes this research with potential limitations and points to some possible outlooks before concluding the paper.

II. RELATED WORK

Numerous approaches have been proposed to accurately identify money-laundering activities [7]. One of the earliest and most common money laundering methods includes the rule engine and the decision tree-based models and predominantly relied on rule-based classification, as shown in fig. 3. Another earlier approach to AML employs transaction monitoring systems that use predominantly rules-based thresholding protocols that are tuned to the volume and velocity of transactions and employ tiered escalation procedures. Rules are a set of logical expressions designed by human domain experts to target a particular fraud problem. Rajput et al. [8] developed an ontology-based expert system to detect suspicious transactions. Although rules are efficient for simple fraud detection, they are inefficient and do not scale in complicated fraud or unknown fraud cases – especially for large-scale graphs involving many nodes. Moreover, rule-based algorithms are easy to evade by fraudsters or weak against adversarial attacks [7].

Other approaches from a graph of money transfers between accounts use a variety of methods ranging from simple logistic regression (LR), support vector machines (SVM), RF, and multilayer perceptron (MLP) to more sophisticated approaches based on GNNs [9]. Several approaches consider structural and behavioural dynamics of dense subgraph detection [4], [7], [10]. Michalak and Korczak [10] used fuzzy matching to capture subgraphs that are more likely to contain suspicious accounts involved in fraudulent activities. The approach proposed by Soltani et al. [4] finds structural similarity-based pairs of transactions with common attributes and behaviours that potentially involve money laundering.

Money laundering often involves high-volume flows of funds through chains of bank accounts between entities [7]. From a transaction graph, many existing approaches, therefore, attempted to detect money laundering by employing structural and behavioural dynamics of dense subgraph detection. However, they do not take into consideration the high-volume flows of funds across bank accounts. Some other approaches tried to assess if the capital flow is involved in money laundering activities using radial basis function (RBF) [7]. However, methods that do not perform flow tracking may yield lower detection accuracy and cannot provide theoretical guarantees. The reason is that the flow across multiple nodes is important for accuracy and robustness against camouflage in money laundering activities [11]. Some recent approaches model the transactions

[3]**GitHub**: https://github.com/rezacsedu/graph_based_aml

in the form of multipartite graphs to detect the complete flow of money from source to destination in an unsupervised manner. FlowScope [7] is a recent flow-based approach, which attempted to detect money laundering behaviour by identifying the chains of transactions w.r.t flows.

Graph analytics techniques offer powerful representations for financial transaction data, whereas graph-based data representations resemble the connectivity of underlying data objects. Graphs offer powerful representations for financial transaction data, whereas graph-based data representations resemble the connectivity of the underlying data objects. Unlike approaches that consider structural and behavioural dynamics of dense subgraphs, graph-based approaches, such as graph neural networks (GNNs) [9] have benefited from their representation learning capabilities (e.g., graph embedding (GE) techniques) from additional graph features. GNNs take into account both the numerical attributes of graph nodes as well as the edges connecting them explicitly. This makes GNNs an effective means to extract complex patterns of interactions between them as shown in fig. 4. This can result in an additional accuracy lift incorporating graph features into traditional ML models [6].

Graph convolutional networks (GCN) [9] are applied for financial crime detection in transaction networks involving money laundering detection [5], [12], phishing detection on the Ethereum blockchain [13], detection of fraudulent transactions [14], and for detecting patterns associated with financial crimes [15]. In particular, in a recent study, Altman et al. [2] have outlined that a lower illicit ratio and a longer period of laundering patterns scenarios pose greater challenges in terms of true detection of money laundering because. They showed that GNN models can recognize laundering patterns in extremely imbalanced multi-graph datasets. Subsequently, graph analytics techniques on graphs have emerged as an increasingly effective means for AML. Another GNN architecture called graph substructure network (GSN) [16] is proposed that can take advantage of pre-calculated subgraph pattern counts to improve the expressivity of GNNs. Further, since large transactional graphs contain billions of nodes and edges, approaches like GraphSAGE [17] and FastGCN [18] exhibit both efficiency and scalability when it comes to graph representation learning [18]. Since transaction graphs are often large, scalability is crucial. Therefore, recent approaches, e.g., FastGCN achieved high accuracy on large benchmark datasets while outperforming GCN and GraphSAGE by up to two orders of magnitude [5].

In low-labelled data scenarios, unsupervised techniques can learn low-dimensional representations of nodes by leveraging graph structures and features. Semantic Web (SW) technologies address data variety and offer a unifying data model by which transaction data can be mapped in a graph structure called knowledge graphs (KGs) [19]. A KG can be defined as $G = \{E, R, T\}$, where G is a labelled and directed multi-graph, and E, R, T are sets of entities, relations, and triples, respectively. A triple in $\mathcal{G}$ can be formalized as $(h, r, t) \in T$, where $h \in E$ is the head node, $t \in E$ is the tail node and $r \in R$ is the edge connecting h and t(i.e., relation r holds between h and t) [20]. Nodes in a KG represent entities and edges represent binary relations between entities. The effectiveness of a GE model depends on its capability to learn useful representations of the nodes that play a significant role in any downstream learning tasks such as link prediction.

A GE technique aims to embed entities and relations in a KG into a low-dimensional dense feature space while preserving its properties [21]. GE models involve three steps: entity and relation representation, scoring function definition, and learning entity and relation representation [21]. Translation embedding methods, e.g., TransD and TransE [22] create embeddings by representing relations as translations from a head entity to a tail entity [23]. Embeddings are optimized w.r.t proximity measure $h \bigoplus r \approx t$ to preserve the relationship in the graph. The resulting embeddings of entities and relations provide denser representations of the domain, making them suitable for a variety of downstream tasks. However, most translation embedding methods have limited capacity in modelling complex relations [22]. To address these shortcomings, GraphSAGE [17] is proposed, which learns embeddings of unlabeled nodes by utilizing the graph structure and node features. GraphSAGE can extract embeddings of unseen nodes without requiring retraining. Unlike Node2Vec which learns a lookup table of node embeddings, GraphSAGE learns a function that generates embeddings by sampling and aggregating attributes from each node's local neighbourhood and combining those with node's attributes [17].

Transaction graph data often have a complex temporal dependency, where historical transactions have an impact on current transactions, e.g., transactions have complex spatial correlation [24]. However, the majority of GE models take into consideration only spatial information thereby ignoring temporal information, even though each transaction has an associated timestamp. Spatio-temporal variants of GNN [25] are employed in several applications starting from predictive learning in urban computing [26] to money laundering fraud detection [24]. EvolveGCN [27] is proposed to extract node embeddings by integrating both spatial and temporal information. EvolveGCN uses a recurrent neural network (RNN) to evolve the parameters of GCN along the temporal axis, providing flexibility for modelling temporal data without relying on node embeddings.

Attention-based architectures that are employed by leveraging attention mechanisms also show potential when dealing with arbitrarily structured graphs. Following a self-attention strategy, hidden representations of each node in the graph are computed by attending to its neighbours. One particular approach is called dynamic graph transformer (DGT) [28]. It consists of two modules: *tansformer* module that captures cross-domain knowledge using attention mechanism, and the *pooling* that generates informative node embeddings using

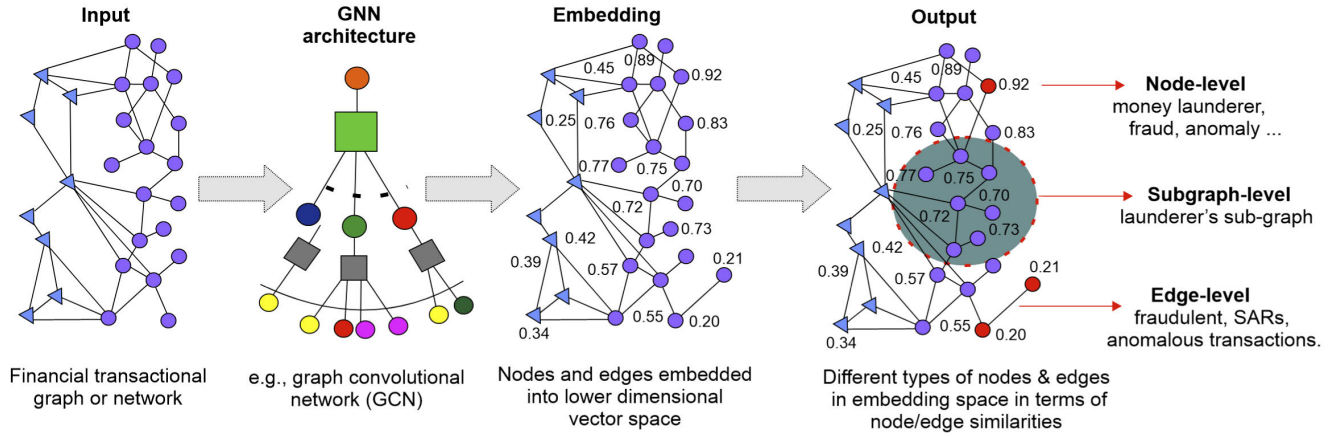


FIGURE 4. General workflows of using GNNs for anomaly detection in financial transactional graphs.

the final attention layer. Further, attention architectures have several interesting properties. For example, the attention mechanism is efficient as it is parallelizable across node-neighbour pairs, it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbours, and the underlying model can be applied to inductive learning problems [2]. Further, attention-based approaches are generalizable to unseen graphs. Subsequently, they are for node classification [2].

III. METHODS

We employ semi-supervised graph learning techniques on transaction graphs to identify nodes involved in potential money laundering. We employ both pipeline and *end-to-end* approaches as outlined in fig. 5 and fig. 6, respectively. In the pipeline setting, an embedding model is first trained to generate node embeddings that are used, along with additional node features, to train binary classifiers along with other local graph features. For the latter, the node classification is performed in an end-to-end setting, without having to train a classifier.

A. PROBLEM FORMULATION

We employ semi-supervised learning that uses annotations from alert data and embeds graph nodes into a lower-dimensional vector space, which are used to train a binary classifier to predict the suspiciousness of a node w.r.t its direct or indirect connections to nodes that are known to be suspicious. To apply semi-supervised learning, we randomly remove a certain percentage of nodes from the graph along with all edges connected to them. Next, we train a GE model on the remaining sub-graph. During inference, we generate embeddings for the removed nodes using the trained GE model that is used to predict the labels of the held-out nodes once they are re-inserted back into the network.

Let a graph $\mathcal{G} = (V, E)$ where V represents accounts and E represents transfers. We split V into three sets: X and Y, which contain the outer accounts with net transfers into and out of the bank, respectively, and W, which contains the inner accounts of the bank. For any $v_i, v_j \in V$ where $(i, j) \in E$, e_{ij} denotes the amount of money transferred from account v_i to v_j. Given this setup, we can represent a directed KG as a set of triplet facts $(h, r, t) \in F$ such that $\mathcal{G} = (V, E, F)$ denotes a link $r \in R$ from the head $h \in V$ to the tail $t \in V$. Let Γ be the embedding model that maps each node v_i of the graph to a vector $\vec{v_i} \in \mathbb{R}^d$, where d is the dimension of the embeddings and N is the number of nodes. This embedding Γ captures the information of the graph and is used to generate a set of vectors $\vec{V}$ for all nodes. It is to be noted that depending on the embedding method Γ and embedding dimension, different embedding vectors can be generated for the entities. The task is then training a classifier f on $\vec{V}$ to predict if a node is of suspiciousness, where the prediction $\hat{y}_i$ for embedding vector $\vec{v_i}$ for the i^{th} node can be defined as follows:

$$\hat{y}_i = f(\vec{v_i}) = \begin{cases} 1, & \text{if flagged, e.g., SAR or illicit} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

B. GENERATING DIRECTED GRAPHS

We generate a directed transactional graph from transactions, alerts (e.g., transactions containing flags SARs/illicitness), and party datasets, followed by preprocessing and encoding categorical features. Then, we generate nodes and edges that are used to form a directed graph. Finally, we annotate the nodes with alert datasets or additional features (i.e., labelled SARs indicating whether a node was involved in any of the known money laundering schemes or illicit).

C. GRAPH EMBEDDINGS

Since ML classifiers require fixed-length vectors as input, we employ different unsupervised graph representation learning techniques such as Node2vec, Attri2Vec, GraphSAGE, and DGT models to generate node embeddings. They represent the neighbourhood of a node and their relations to the neighbouring nodes. Using *Node2Vec*, a corpus of text $\mathcal{C}$ is generated by performing uniform random walks starting from each entity in the graph [29]. Then, $\mathcal{C}$ of edge-labelled

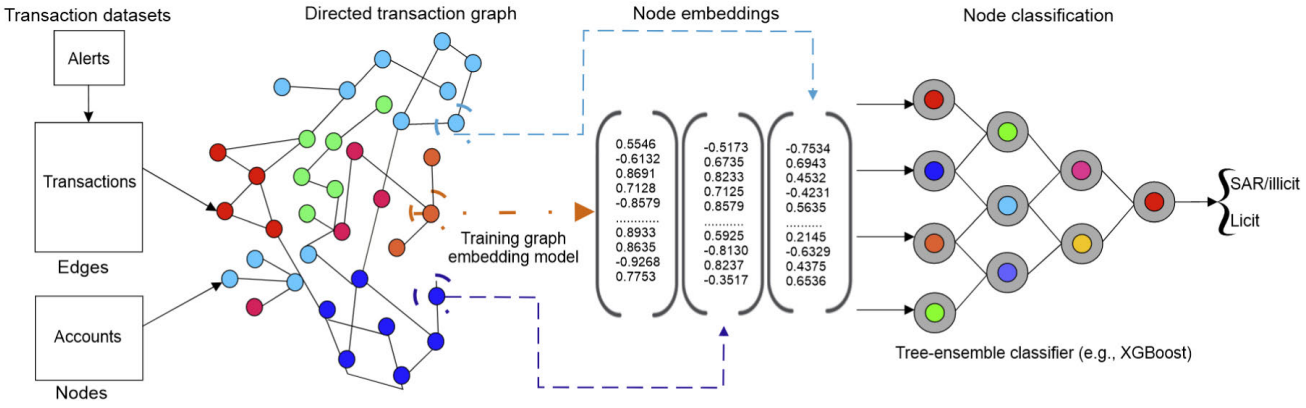


FIGURE 5. Workflow of pipeline methods for identifying money laundering nodes.

random walks are used as the input for learning embeddings of each node using skip-gram (SG)-based Word2vec [30] model. From a given a sequence of facts $(w_1, w_2, \ldots, w_n) \in \mathcal{C}$, SG aims to maximize the average log probability L_p according to the context within the fixed-size window [30]:

$$L_p = \frac{1}{N} \sum_{n=1}^{N} \sum_{-c \le j \le c, j \ne 0} \log p\left(w_{n+j} | w_n\right), \tag{2}$$

where c is a context. Negative sampling is used to set $p\left(w_{n+j}|w_n\right)$ by replacing $\log p\left(w_O|w_I\right)$ with a function to discriminate the target (w_o) and by drawing k words from a noise distribution $P_n(w)$ as follows [30]:

$$\log \sigma\left(v'^{\top}_{w_O} v_{w_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma\left(-v'^{\top}_{w_i} v_{w_I}\right)\right]. \tag{3}$$

The embedding of a concept c occurring in corpus $\mathcal{C}$ is the vector $\vec{v}_s$ in eq. (3) derived by maximizing eq. (2). Both Word2vec and Node2Vec algorithms follow a 2-step representation learning technique. Step 1 involves the use of second-order random walks to generate sentences from a graph, where a sentence is a list of node IDs. A corpus (the set of all sentences) is then used to learn an embedding vector for each node in step 2. Each node ID is considered a unique token in a dictionary having a size of number of nodes N in $\mathcal{G}$. Attri2Vec [31] is trained to learn node representations with non-linear mapping on node content attributes. To capture structural similarity in learned node representations, Attri2Vec employs DeepWalk to make nodes sharing similar random walk context nodes represented closely in the subspace. For each (target, context) node pair $\left(v_i, v_j\right)$ from random walks, Attri2Vec learns the representation $\vec{v}_i$ for the target node v_i by using it to predict the existence of context node v_j using a three-layer network. The representation of a node $\vec{v}_i$ in the hidden layer is obtained by multiplying its raw feature vector in the input layer with the input-to-hidden weight matrix W_{in} [31].

For a large set of "positive" (target, context) node pairs from random walks and an equally large set of "negative" node pairs randomly selected from $\mathcal{G}$ according to a certain distribution, GraphSAGE learns a binary classifier that predicts whether arbitrary node pairs are likely to co-occur in a random walk on graph [17]. GE models we consider so far take into consideration only spatial/structural information, thereby ignoring temporal information. We learn knowledge from such a dynamic graph, with the hypothesis that the AML could benefit from it since DGT can capture both spatial and temporal information simultaneously [28].

Let node v_1^t and v_2^t be involved in a transfer at time t, where their common connections had multiple transactions in previous timestamps. This temporal relation can be modelled as $u_1^{t-1} - u_2^{t-1}$ and $u_1^{t-2} - u_2^{t-2}$. To extract spatial-temporal knowledge, node encodings are aggregated within a substructure node set into node embeddings. Attention is applied to exchange information across nodes. An attention layer is represented as [28]:

$$\mathbf{H}^{(l)} = \text{att}\left(\mathbf{H}^{(l-1)}\right) = \text{softmax}\left(\frac{\mathbf{Q}^{(l)}\mathbf{K}^{(l)\top}}{\sqrt{d}}\right)\mathbf{V}^{(l)}, \tag{4}$$

where $\mathbf{H}^{(l)}$ and $\mathbf{H}^{(l-1)}$ is the output embedding for the l and $(l-1)^{th}$ layer, respectively; d is the dimension of node embedding, *att* signifies the self-attention operation; $\mathbf{Q}^{(l)}, \mathbf{K}^{(l)}, \mathbf{V}^{(l)} \in \mathbb{R}^{(\tau(k+2))\times d}$ are query-, key-, and value matrices for feature transformation and information exchange represented as [28]:

$$\begin{cases} \mathbf{Q}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}_Q^{(l)}, \\ \mathbf{K}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}_K^{(l)}, \\ \mathbf{V}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}_V^{(l)}, \end{cases} \tag{5}$$

where $\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)} \in \mathbb{R}^{d\times d}$ are the learnable parameter matrices of the l-th attention layer. In an attention layer, $\mathbf{Q}^{(l)}$ and $\mathbf{K}^{(l)}$ calculate the contributions of different nodes' embeddings, while $\mathbf{V}^{(l)}$ projects the input into a new feature space that is combined as of eq. (4) to acquire the output embedding of each node by aggregating the information of all nodes adaptively [28]. Input to transformer $\mathbf{H}^{(0)}$ represents an encoding matrix of the target edge $\mathbf{X}\left(e_{\text{tgt}}^t\right)$ by setting

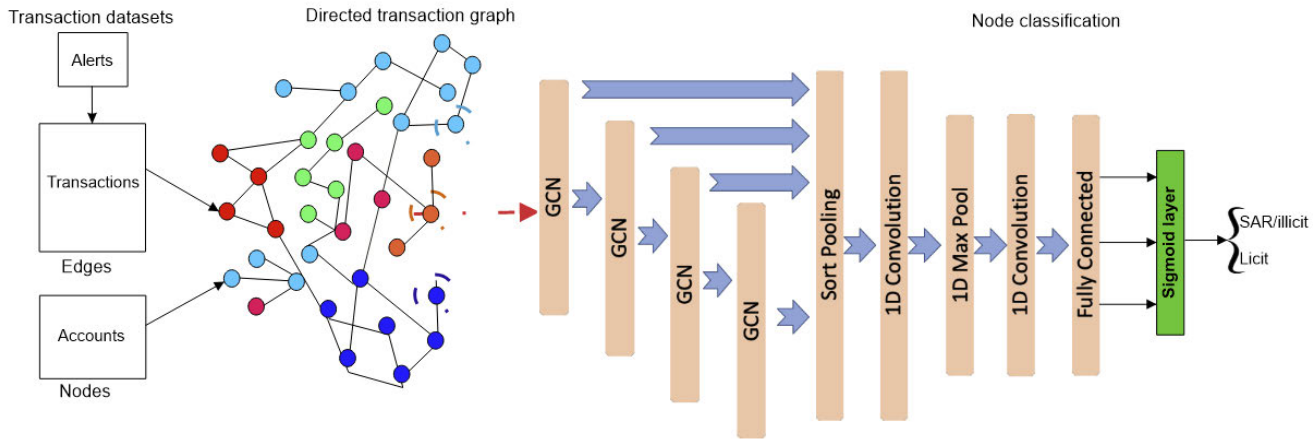


FIGURE 6. Workflow of end-to-end approach based on GCN for identifying money laundering nodes.

$d = d_{enc}$. The output of the final attention layer $\mathbf{H}^{(L)}$ is extracted as the output node embedding matrix $\tilde{\mathbf{Z}}$, where each row represents a node embedding vector.

D. COMPUTING GLOBAL TOPOLOGICAL FEATURES

Global topological features in a directed graph such as shortest paths, centrality, communities, in-flow/out-flow, and triangle counts may provide useful signals, especially communities within a graph, where suspicious parties within the same community might be more likely to be globally suspicious too. Such features may help ML models perform better in financial crime detection. We employ the Dijkstra shortest-path algorithm as the means to calculate how close is a party of interest to a party that has exhibited suspicious behaviour in the past. For a directed graph G, the shortest path from a node u to another node v is computed as [32]:

$$d(u) = \min_{v \in N^+(u)} d(v) + w(u, v), \tag{6}$$

where $d(u)$ is the shortest distance from the source node to node u, $N^+(u)$ is the set of out-neighbors of node u, and $w(u, v)$ is the weight of the edge from node u to node v. We employ the PageRank as the centrality algorithm which measures the influence of a specific graph node on other nodes. We hypothesize that page ranks of individual nodes can be a useful feature for the classifiers – especially when combined with the information if the influential parties were known to be suspicious for known or past money laundering.

For a directed graph G, the PageRank of a node is calculated as the probability of a random surfer landing on that node after following a series of links [33]:

$$PR(u) = \frac{1-d}{N} + d \sum_{v \in N^-(u)} \frac{PR(v)}{k_v^+}, \tag{7}$$

where $PR(u)$ is the PageRank of node u, d is the damping factor, N is the number of nodes in the graph, $N^-(u)$ is the set of in-neighbors of u, and k_v^+ is the out-degree of node v. We employ weakly connected components (WCCs) for the community detection. WCss are used to determine groups of nodes sharing common characteristics or heavily interacting with each other within G. The set of WCCs for G is computed as [32]:

$$WCC(G) = C \subseteq V(G) \mid \forall u, v \in C, u \rightsquigarrow v \vee v \rightsquigarrow u, \tag{8}$$

where $WCC(G)$ is the set of WCCs and $V(G)$ is the set of vertices for G, and $u \rightsquigarrow v$ signify the existence of directed path from u to v. The underlying properties of a community such as its size, the number of transactions within the community, how many parties within the community have had an SAR filed on them, and who are the most influential parties within the community are calculated as additional features for the classifiers.

In a directed graph G, a triangle is formed by a directed cycle of length three, and each vertex in the cycle has a reciprocal edge with another vertex in the cycle. Triangles can help detect anomalies or outliers in G, e.g., if a node has a much higher or lower triangle count than its neighbours, it might indicate that it is behaving abnormally or differently from the rest G [34]:

$$T(G) = \frac{1}{3} \sum_{u \in V(G)} T(u), \tag{9}$$

where $T(G)$ is the total number of triangles in G, $V(G)$ is the set of vertices, and $T(u)$ is the number of triangles that pass through vertex u. Term $T(u)$ in the above formula is calculated as follows [34]:

$$T(u) = \sum_{v \in N^+(u)} d^{\leftrightarrow}(v), \tag{10}$$

where $N^+(u)$ is the set of out-neighbors of u and $d^{\leftrightarrow}(v)$ is the number of reciprocal edges incident to v.

E. TRAINING OF CLASSIFIERS

We train RF, LightGBM, and XGBoost ensemble models on learned embeddings. Internal nodes in a decision tree (DT)

represent feature values w.r.t boolean conditions and leaf nodes represent predicted labels. From a set of embedding vectors $\bar{V}$, a DT iteratively splits X^* into multiple subsets w.r.t to threshold values of features at each node until each subset contains instances from one class only. Each branch in a DT represents a possible outcome, where the interaction between prediction $\hat{y}_i^*$ and feature $\vec{x}_i^*$ is $\hat{y}_i^* = f\left(x_i^*\right) = \sum_{j=1}^{N} c_j I\left\{X_i^* \in R_j\right\}$, R_j is the subset of the data representing the combination of rules at each node, and $I\{.\}$ is an identity function [35].

In the case of tree ensemble models, the prediction function $f(\mathrm{x}^*)$ is the sum of individual feature contributions plus the average contribution for the initial node for the dataset and K possible class labels that change along the prediction path w.r.t objective function (e.g., gini impurity or entropy) causing the split [35]:

$$f(x) = c_{full} + \sum_{k=1}^{M} \sigma(x, k), \tag{11}$$

where c_{full} is the average of the entire training set X dataset (initial node), M is the total number of features. For graph-based node classification, which is technically predicting the label of a node u at time t, we follow the usual training procedures for EvolveGCN [27] and FastGCN [18], followed by the standard GCN-like approach: the activation function of the last graph convolution layer is set to sigmoid so that h_t^u is a probability vector over two probable classes (i.e., licit and illicit).

IV. EXPERIMENTS

In this section, we report our evaluation results.

A. DATASETS

We evaluated our approach on four datasets: *AMLSim*, *Elliptic*, *IBM AML*, and *SynthAML* [36]. The *IBM AML* is a synthetic financial transaction dataset generated using an agent-based generator, followed by calibrated to match real transactions as closely as possible. This dataset has 6 versions that are divided into two groups of three: i) group HI has a relatively higher illicit ratio, and ii) group LI has a relatively lower illicit ratio. Both HI and LI internally have three sets of data: small, medium, and large. All these datasets are independent, e.g. the small datasets are not a subset of the medium ones. For our study, only the HI-medium version is used with a focus on node classification (given it has higher illicit ratios), leaving the LI version for future study. The SynthAML is also a synthetic dataset for benchmarking statistical and machine learning (ML) methods for AML. It employs Synthetic Data Vault (SDV) to tune a probabilistic model with real data from Danish bank *Spar Nord* with approx. 440,000 clients.

SynthAML contains 20,000 AML alerts and over 16 million transactions in two tables. The alert dataset has an alert ID, the date the alert was raised, and the outcome of the alert.[4] Each transaction has four features: i) a transaction timestamp, ii) the transaction entry (credit vs. debit), iii) the transaction type (card, cash, international, or wire), and iv) the transaction size (measured in log Danske Kroner and standardized to have zero mean and unit variance). The AMLSim dataset, which is generated with a multi-agent simulation platform is tailored for an AML problem. Each agent behaves as a bank account transferring money to other agent accounts in which a few agents conduct nefarious activity modelled on real-world patterns. We generate a dynamic directed transaction graph containing semi-realistic suspicious activities, based on the following information and graph generation process:

- **Accounts**: whose transactions are monitored.
- **Alerts**: transactions that are frequently or periodically monitored and triggered alerts (illicit or SARs) according to AML guidelines.
- **Transactions**: list of all transactions (both normal and SARs) including sender and receiver accounts.

Each node represents an account that has an account number, account type, owner name, and date/time created. Nodes are designated with `cash in`, `cash out`, `debit`, `payment`, `transfer` or `deposit` activities and are of either organization or individual types. Each edge has a transaction ID, amount, and timestamp. Data is sparsely labelled with flagged transactions (e.g., transactions that violate volume and velocity rules) and SARs (e.g., transactions that are confirmed for suspiciousness). Elliptic dataset[5] is a graph network of Bitcoin transactions with handcrafted features constructed using publicly available information. This anonymized data set is a transaction graph collected from the Bitcoin blockchain. The dataset maps Bitcoin transactions to real entities in two categories: licit and illicit [37]:

- **Licit**: licit transactions contain usual exchanges, wallet providers, miners, licit services, etc.
- **Illicit**: illicit transactions contain scams, malware, terrorists, ransomware, Ponzi schemes, etc.

The graph contains 203,769 node transactions and 234,355 directed edge payments flow out of which 2% are illicit and 21% are licit. The remaining 77% of samples are labelled as *unknown* transactions. Each node has 166 features associated, where the first 94 features represent local information (i.e., time-step, number of inputs/outputs, transaction fee, output volume and aggregated figures, e.g., average BTC received and spent by inputs and outputs, the average number of incoming and outgoing transactions). The remaining 72 features represent aggregated features that are obtained using one-hop backwards/forwards transaction information from the centre node (i.e., min/max standard deviation and correlation coefficients of neighbour transactions w.r.t number of inputs/outputs, transaction fee, etc.). Time steps are associated with each node, representing an estimated time when the transaction is confirmed. 49 timesteps are evenly spaced with an interval of 2 weeks.

[4] i.e. if the alert was reported to the authorities or dismissed.

[5] https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

Using the SynthAML dataset, we generate a directed transaction graph similar to the AMLSim dataset. Then, similar to Jensen et al. [36], we focus on node classification based on synthetic data only, where we classify alerts based on their outcomes. As for the training set, alerts raised between January 1, 2020 and December 31, 2020, were used. As for the test set, we used alerts raised between January 1, 2021 and December 31, 2021. However, unlike using statistical graph features like min, mean, median, max, standard deviation, counts, and sum per transaction type and entry for all transactions associated with each alert, which yields as many as $7 \times 2 \times 4 = 56$ features per alert. As for the IBM AML dataset, we split the transaction indices after ordering them w.r.t their timestamps similar to [2], where the data split is defined by two timestamps: t_1 and t_2. Train set includes transactions before time t_1, the validation set includes transactions between times t_1 and t_2, and the test set includes transactions after t_2. These yield three dynamic graphs at times t_1, t_2, and $t_3 = t_{\max}$.

B. EXPERIMENT SETTINGS

First, each dataset is subset chronologically for training, validation, and testing (e.g., 60%, 20%, and 20% splits or temporal splits w.r.t associated timestamp). As for the pipeline methods, Node2Vec, Attri2Vec, GraphSAGE, and DGT models are trained to generate node embeddings that are used to train the classifiers.[6] As for the end-to-end approach, SkipGCN [9], EvolveGCN [27], and FastGCN [18]. SkipGCN and FastGCN are trained in batch to reduce training costs through neighbourhood sampling, while EvolveGCN is trained to capture the dynamism by evolving GCN parameters using the AdaGrad optimizer with varying learning rates and batch sizes. Schlichtkrull et al. [6] show that if there are more than three convolutional layers, node features will be over-smoothed, e.g., all the nodes on the graph look similar. Inspired by this, we configured GCN, FastGCN, and EvolveGCN to have three convolutional layers. Further, since classes are imbalanced in both datasets, we trained SkipGCN, FastGCN, and EvolveGCN using *weighted cross-entropy loss* to provide higher importance to minority class (i.e., illicit/SARs).

We used open-source StellarGraph library[7] for computing node embeddings, which is based on BiasesRandomWalk and Word2Vec from the Gensim library. Attri2Vec was trained using a skip-gram model with a window size of 5, a graph walk depth of 5, and 500 walks per entity. DGT model was trained[8] by varying the number of layers L between [1,5]. While generating graph-based features, the damping factor for PageRank is set between [0.75 and 0.85]. For the sake of semi-supervised learning, we randomly removed 10-20% of the nodes from each experiment and trained the GE models on the reduced sub-graph. We generated embeddings for the removed nodes using the trained GE model during inference and used these embeddings to predict the labels of the originally held-out nodes after re-inserting them back into the network. Since our semi-supervised learning techniques use both labelled and unlabeled data, a model can suffer from overfitting, especially when the labelled data is scarce or noisy, or when the unlabeled data is not representative of the target distribution. To mitigate this, we employ two strategies: aside from *l1*, *l2*, and *dropout* regularization, we employ consistency regularization using *Mean Teacher* which enforces the model to learn only smooth and robust features during node embeddings. Second, we employ a self-training technique to iteratively label the unlabeled data using the model's predictions, followed by adding the most confident ones to the training set.

The open-source implementations were used to train the RF,[9] LightGBM,[10] and XGBoost[11] ensemble classifiers. The best hyperparameters were selected via random search and 5-fold cross-validation settings. We evaluated the performance of each trained classifier w.r.t area under the precision-recall curve (AUPR), Matthews correlation coefficient (MCC), and F1 scores.

C. ANALYSIS OF NODE CLASSIFICATIONS

Table 1 summarizes the results of the prediction task based on pipeline methods. As for the pipeline methods, the combination of DGT and XGBoost outperforms all other combinations across datasets. The results also reveal that each classifier performs worse when trained on embeddings generated by the Node2Vec and Attri2Vec models. Although we observed slightly different results for the IBM AML and SynthAML datasets, the overall trend is similar. When our pipeline approach was tested on *SynthAML* [36] dataset, it turned out that the best XGBoost classifiers significantly outperformed their best classifiers RF and LightGBM, showing an F1-score of 0.726 against the same score of 0.64. Besides, the DGT + XGBoost slightly outperformed the GNN-based models when tested on the IBM AML HI-Medium version.

On the other hand, end-to-end methods outperform every pipeline method, as marked in green in tables table 2 and 3. The EvolveGCN model outperforms all pipeline methods with tree-ensemble classifiers, indicating the effectiveness of end-to-end methods compared to their pipeline counterparts. Moreover, EvolveGCN consistently outperforms both Skip-GCN and FastGCN, although the improvement is not very substantial.

When it comes to comparative analysis between pipeline and end-to-end methods, the performance of GraphSAGE and XGBoost is comparable to that of DGT + XGBoost. However, when local and global topological features were

[6] **GitHub**: https://github.com/rezacsedu/graph-based-aml
[7] https://github.com/stellargraph/stellargraph
[8] https://github.com/yuetan031/TADDY_pytorch
[9] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html/
[10] https://lightgbm.readthedocs.io/en/stable/
[11] https://XGBoost.readthedocs.io/

TABLE 1. Node classification results for pipeline methods.

		AMLSim			Elliptic			IBM AML			SynthAML		
GE model	Classifier	AUPR	F1	MCC	AUPR	F1	MCC	AUPR	F1	MCC	AUPR	F1	MCC
Node2Vec	RF	0.751	0.760	0.651	0.806	0.817	0.622	0.637	0.647	0.529	0.661	0.643	0.547
	LightGBM	0.753	0.760	0.649	0.800	0.813	0.623	0.640	0.651	0.528	0.662	0.640	0.536
	XGBoost	0.806	0.801	0.752	0.885	0.874	0.673	0.642	0.649	0.531	0.668	0.649	0.552
Attri2Vec	RF	0.775	0.782	0.669	0.821	0.832	0.665	0.647	0.656	0.546	0.668	0.641	0.545
	LightGBM	0.769	0.791	0.657	0.823	0.832	0.659	0.651	0.657	0.547	0.669	0.643	0.542
	XGBoost	0.806	0.801	0.752	0.902	0.894	0.673	0.652	0.658	0.551	0.671	0.652	0.549
GraphSAGE	RF	0.802	0.804	0.675	0.891	0.882	0.778	0.667	0.668	0.567	0.683	0.692	0.563
	LightGBM	0.794	0.809	0.685	0.883	0.874	0.750	0.669	0.671	0.568	0.682	0.688	0.557
	XGBoost	0.815	0.816	0.701	0.912	0.905	0.782	0.669	0.671	0.569	0.694	0.687	0.576
DGT	RF	0.813	0.825	0.693	0.907	0.897	0.753	0.677	0.675	0.576	0.687	0.694	0.581
	LightGBM	0.805	0.811	0.677	0.894	0.862	0.733	0.679	0.680	0.579	0.689	0.691	0.585
	XGBoost	**0.833**	**0.832**	**0.715**	**0.918**	**0.915**	**0.792**	**0.683**	**0.684**	**0.582**	**0.712**	**0.713**	**0.593**
	XGBoost*	**0.852**	**0.846**	**0.727**	**0.925**	**0.918**	**0.802**	**0.725**	**0.7014**	**0.623**	**0.727**	**0.726**	**0.618**
	XGBoost†	**0.864**	**0.859**	**0.738**	**0.939**	**0.932**	**0.814**	**0.736**	**0.712**	**0.632**	**0.738**	**0.737**	**0.627**

added along with node embeddings, we observed noticeable improvements in pipeline methods that we discuss in section IV-F.

D. ANALYSIS OF NODE EMBEDDINGS

One of the factors that we explored in our experiments was the embedding dimension, which is the number of features used to represent each node in the graph. We tested different values of d for each GE model, ranging from 32, 64, 128, 256, 300. We then visualized the node embeddings in a low dimensional space using t-SNE, as shown in fig. 7. The plots reveal that there are clear differences between the embeddings of normal and illicit classes, meaning that the embeddings capture some information that can help identify fraudulent or SAR accounts/nodes. Moreover, we observe that the embeddings generated by GraphSAGE and DGT models are more clustered and separated than those produced by Node2Vec and Attrib2Vec models. This indicates that GraphSAGE and DGT models learn more meaningful and discriminative representations of the nodes, which can benefit a classifier in detecting fraud and nodes.

E. EFFECTS OF TEMPORAL INFORMATION

In table 1, we present the results of applying the XGBoost model on all datasets. We compare the performance of the XGBoost model when it uses different types of node features as input. We show that the performance of the XGBoost model is improved when it uses the temporal information that is learned by the DGT model, which is highlighted in *grey* in the table. This indicates that the DGT model can capture the temporal dynamics of the graph and generate more informative and discriminative features for the nodes. The XGBoost model that uses the DGT features outperforms other pipeline methods that use static or semi-static features, such as Node2Vec, Attrib2Vec, or GraphSAGE.

We also observe that the combination of GraphSAGE and XGBoost model achieves the best performance on the Elliptic dataset, which suggests that the GraphSAGE model can leverage the node attributes and the graph structure to generate more expressive and relevant features for the nodes. We attribute the success of these dynamic models to the fact that the input features contain useful information about the nodes and their relationships and that the transformers can learn more abstract and higher-level features that are crucial for distinguishing between normal and illicit nodes. These results also demonstrate that the representation learning capability of these models is dependent on the quality of the input features and that the learned representations are reflected in the classification outcomes.

F. ANALYSIS OF COMBINED FEATURES' EFFECTS

In this section, we analyse how the node embeddings obtained from different models can be combined with other types of features to enhance the node classification performance. Pareja et al [27] showed that adding aggregated information to the original inputs, such as the node degree or the node label, can improve the F1 scores for anomaly detection tasks. Motivated by this idea, we experimented with different combinations of node embeddings, local node features, global topological features, and original input space, and we retrained individual classifiers on the extended feature space. We show examples for the AMLSim dataset, where we used the local node features and global topological features. We compared the results with the baseline methods that use only node embeddings or only original inputs as features. Figure 8 illustrates the effects of combining node embeddings and other features such as local graph features, and topological graph features on all datasets: AMLSim, Elliptic, IBM AML, and SynthAML.

We can observe that adding local node features and global topological features to the node embeddings could help slightly increase the classification accuracy, making it similar to end-to-end methods that use graph neural networks. Specifically, we can see that the node classification accuracy with the DGT embeddings + local graph features + XGBoost combination improved (marked in light orange in table 1 and fig. 8a) by 1 to 2% for all datasets compared to the baseline methods. The node classification accuracy further improved by 1.5% for all datasets with the most impactful features (among DGT embeddings, local node features, and

TABLE 2. Node classification results in end-to-end settings: AMLSim and Elliptic datasets.

	AMLSim			Elliptic		
Model	AUPR	F1	MCC	AUPR	F1	MCC
Skip-GCN	0.834 (0.792)	0.915 (0.875)	0.881 (0.763)	0.928 (0.793)	0.916 (0.873)	0.854 (0.763)
FastGCN	0.841 (0.804)	0.927 (0.890)	0.903 (0.781)	0.933 (0.805)	0.925 (0.881)	0.875 (0.781)
EvolveGCN	**0.869** (0.813)	**0.934** (0.902)	**0.891** (0.773)	**0.941** (0.813)	**0.934** (0.891)	**0.891** (0.773)

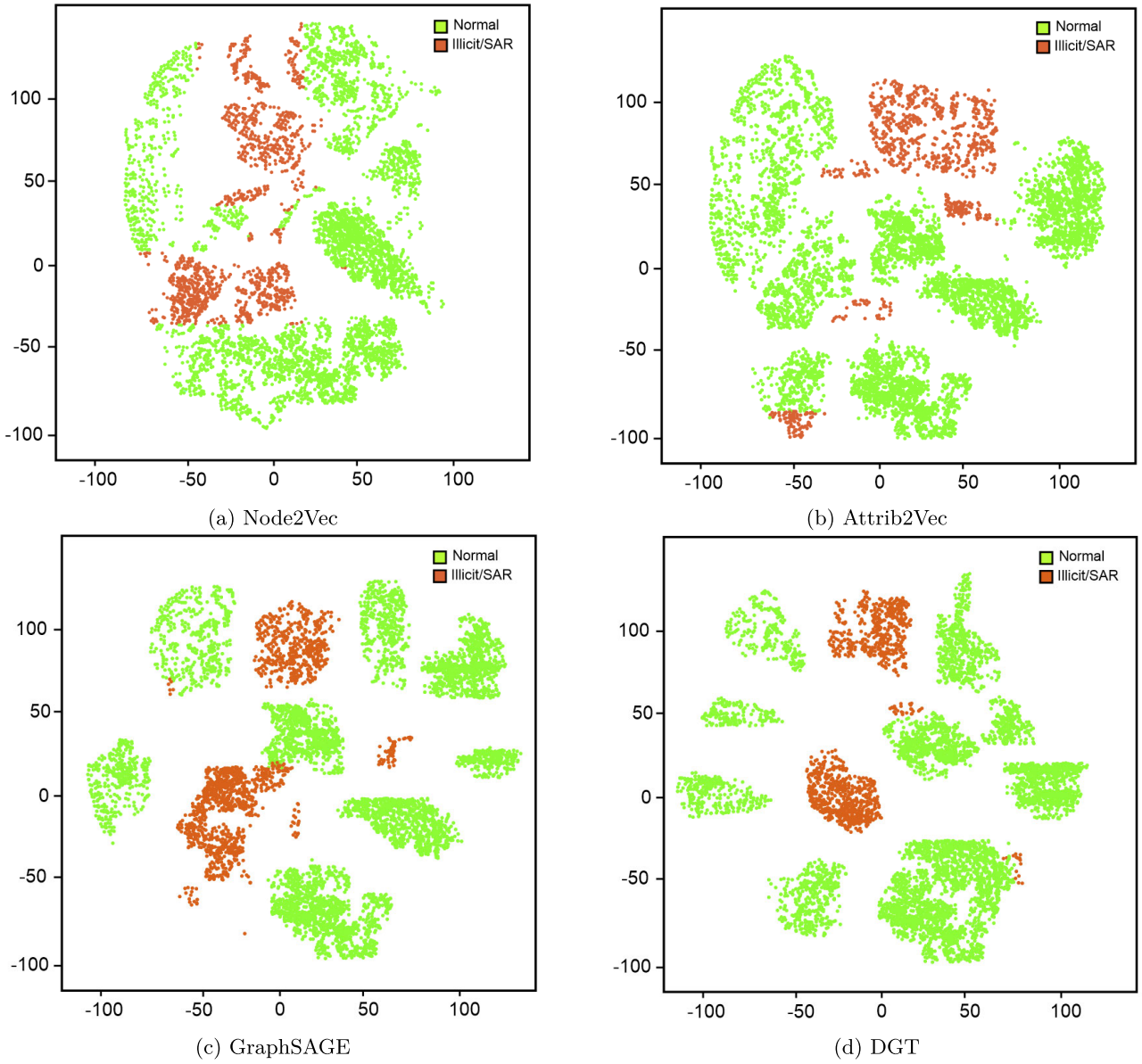
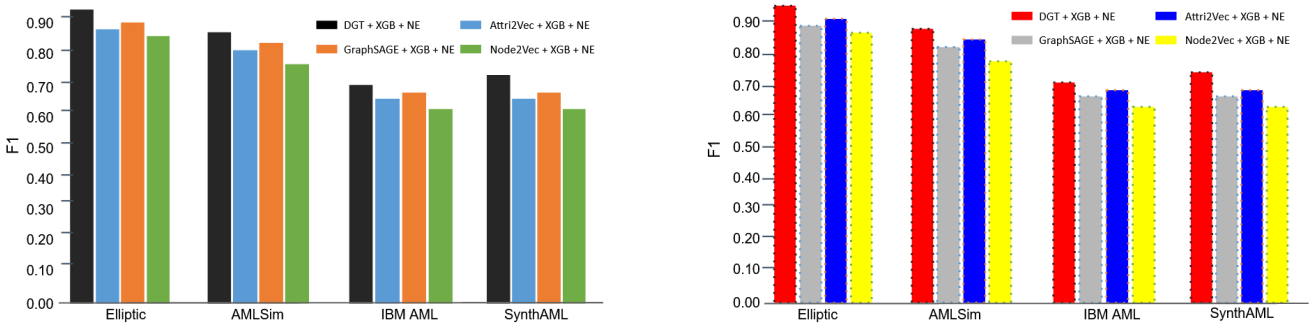


FIGURE 7. t-SNE projection of the node embeddings for the AMLSim dataset into lower dimensional space.

topological features) + XGBoost combination (marked in green in table 1 and fig. 8b), making the node classification results comparable to EvolveGCN, which is the SOTA method for dynamic graphs. These findings suggest that the node embeddings learned by the dynamic models can be enriched by incorporating other features that capture the local and global properties of the nodes and that the extended feature space can help the classifiers distinguish between normal and illicit nodes more effectively.

We also observed for the IBM AML and SynthAML datasets that the encapsulation of node embeddings with the additional node features and full feature space, helped the pipeline methods outperform their end-to-end counterparts. Even though the micro averages for all pipeline



(a) Embeddings + topological features + input features

(b) Node embeddings + impactful features

FIGURE 8. Effects of combining node embeddings, local node features, topological graph features, and most impactful features into original feature space.

TABLE 3. Node classification results in end-to-end settings: for IBM AML and SynthAML datasets.

	IBM AML			SynthAML		
Model	AUPR	F1	MCC	AUPR	F1	MCC
Skip-GCN	0.663	0.664	0.562	0.685	0.677	0.562
FastGCN	0.675	0.678	0.595	0.705	0.692	0.583
EvolveGCN	**0.695**	**0.698**	**0.615**	**0.726**	**0.727**	**0.675**

approaches are above 0.93, they are not very informative for highly imbalanced datasets. Nevertheless, in financial crime forensics, the minority illicit class is of primary interest. Therefore, we report the minority F1 scores for all datasets we experimented with.

G. EXPLAINING MONEY LAUNDERING

The *black-box* nature of a GNN or GE model[12] may raise concerns about transparency and accountability when an AML model is deployed in a real financial system. The latent factors learned by a GNN or GE model are not easily interpretable. Thus, predictions made by such a complex model cannot be traced back, making it unclear how or why they arrived at a certain outcome [19], [38]. On the other hand, disentangling them can provide insights into what features were captured by the representations and relevant for the tasks [19].

Explainable artificial intelligence (XAI) aims to make AI systems more transparent and understandable to humans by interpreting how *black-box* models should make decisions [19]. An interpretable AML model can reveal the factors that impact (e.g., statistically significant features) its outcomes and explain the interactions among them [19], [39]. Thus, an interpretable ML model that emphasizes transparency and traceability of its logic can explain *why and how* it arrived at certain decisions, reducing negative consequences. In the context of our anti-money laundering scenario, *local interpretability* can provide reasons for a decision made for a specific party or reference to similar cases, allowing the identification of unique characteristics of a party or multiple parties in a small group like communities. In contrast, *global interpretability* could show the overall behaviour of an AML model at a high level. While local explanations focus on explaining individual predictions, global explanations explain entire model behaviour.

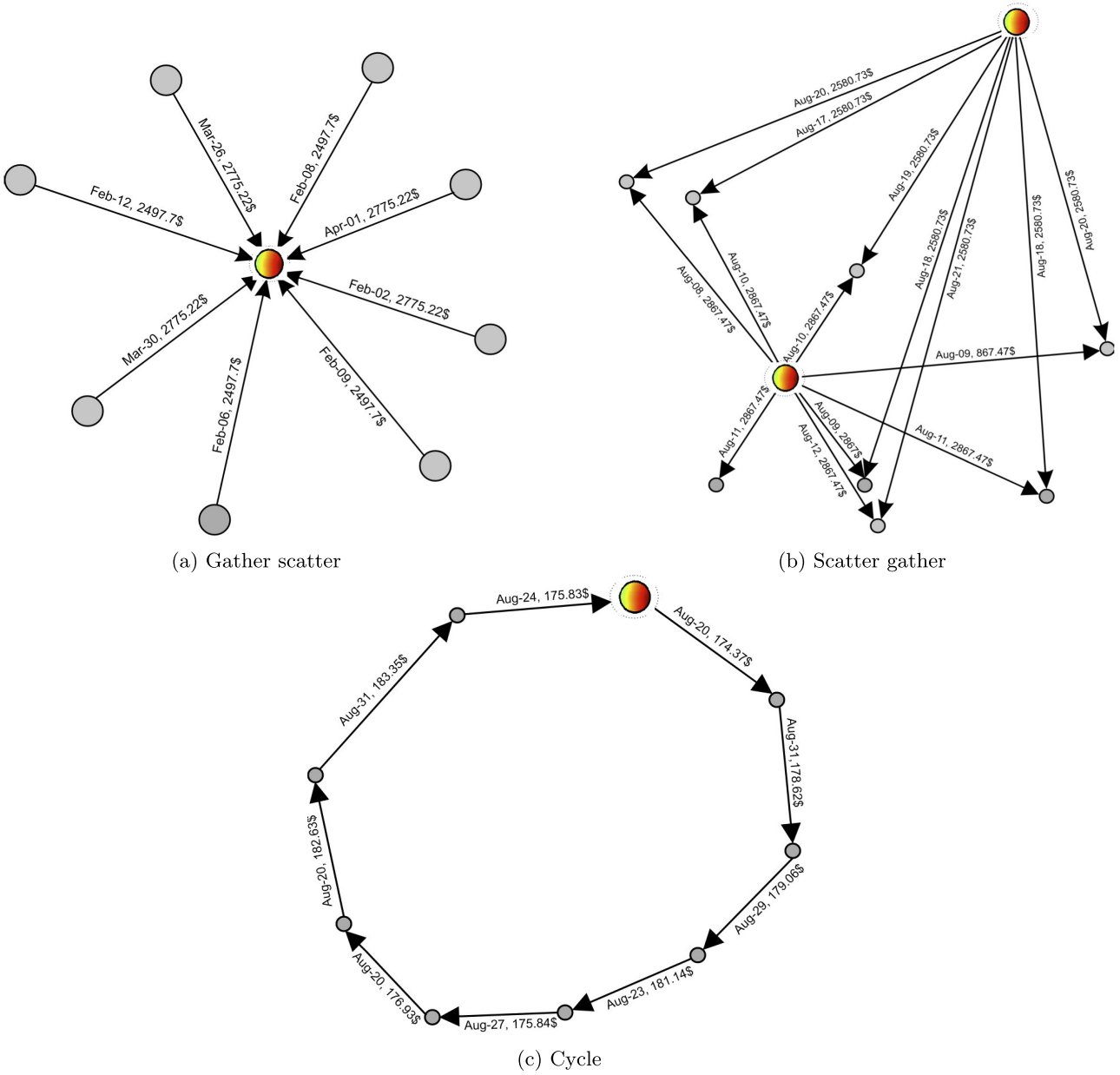
A transaction graph can be viewed as the discrete symbolic representation of knowledge about different types of transactions. Therefore, we employed graph-based explanation techniques such as GNNExplainer [40] and SHAP to compute the contributions of the neighbouring node types and edges when predicting the suspiciousness or involvement of a targeted node/party with money laundering. To provide local explainability, we highlighted some nodes that are flagged with "alert types" such as `gather scattered`, `scattered gather`, and `cycles` for the AMLSim dataset in fig. 9.

As shown in fig. 9a, ttgather scattered is when launderers (node in colour) collect small amounts of cash from various sources[13] and deposit them into a single account or location. This reduces the risk of detection by avoiding large cash transactions that may raise suspicion. As shown in fig. 9b, ttscatter gathered is when launderers (node in colour) transfer the gathered money into multiple accounts across countries or jurisdictions (e.g., offshore accounts, shell companies, or foreign investments). This increases the complexity and anonymity of the money trail, making it harder to follow and recover. As shown in fig. 9c, ttcycles is when launderers (node in colour) repeat the process of gathering and scattering money multiple times, using multiple intermediaries involving different currencies.[14] This further obscures the origin and destination of the money and creates layers of transactions that can confuse or mislead investigators. The feature impact plot for the XGBoost classifier is depicted in fig. 10, showing which features contributed most.

[12]Complex ensemble or DNN models oftentimes tend to be less and less interpretable and may end up as *black-box* methods.

[13]Launderers may gather scattered cash from human trafficking, drugs sales, or tax evasion and deposit into a bank account.

[14]They may cycle money through various banks or individuals using wire transfers or cryptocurrencies.



(a) Gather scatter

(b) Scatter gather

(c) Cycle

FIGURE 9. Different (alter) types of transactions in the AMLSim graph.

As shown, node embeddings, followed by other derived and global topological features for individual nodes such as the shortest path to a node known as SAR, triangle counts, number of communities a node part of, centrality score, PageRank (learned from centrality measures), indegree, outdegree, account type (i.e., individual/private or organizational), transaction type (e.g., cash in/out, transfer, debit, parent, deposit), and amount (e.g., amount in $), are more important than local graph features. It is to be noted that some features such as account number, owner name, transaction ID, etc. are excluded from the original feature space as they do not carry meaningful information for GE or a classifier.

H. AML MODELS IN REAL FINANCIAL SYSTEMS

Since financial fraudulent activities are getting rampant, financial institutions should deploy accurate and robust AML models to satisfy the regulators. Such AML models are expected to exhibit fewer false positives (e.g., alerts that eventually turn into real money laundering) and fewer false negatives (e.g., real money laundering cases that were not detected). False positives are more critical since inaccurate detection will tend human experts to flag too many transactions as illicit, leading to higher human investigation costs (ref. fig. 3). Further, deploying and inferencing money laundering in real-time based on large-scale transaction graphs could be very challenging for many reasons. For

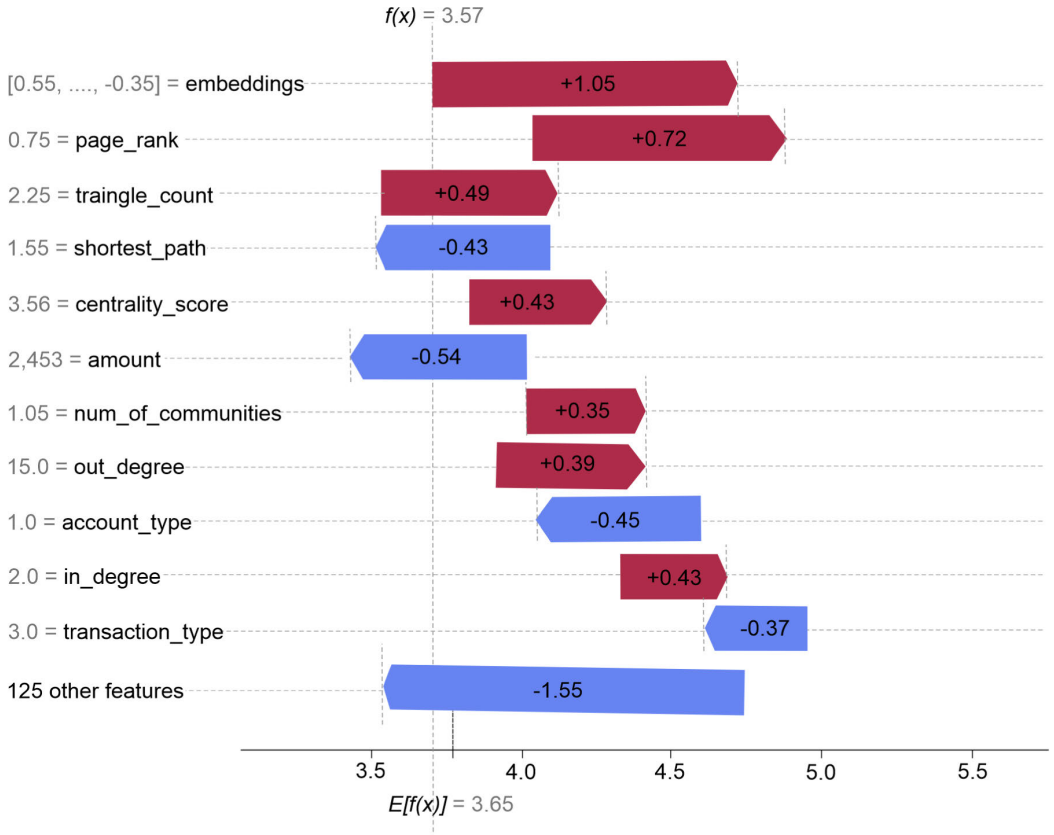


FIGURE 10. Explaining a money laundering example for the AMLSim dataset using the SHAP waterfall plot for the XGBoost classifier. The bottom starts as the expected output value; each row shows how the positive (red) or negative (blue) contribution of individual features pushes the value from the expected output over the training set to model output. Assuming the true class label is illicit, the positive values imply probabilities of > 0.4 that the party is truly involved in money laundering.

example, criminals often mask the true nature of their transactions using complicated account layering or multi-hop transactions. This makes the identification of money laundering a complex problem.

The more trainable parameters an AML model will have, the larger its size will be, making the deployment infeasible for devices with limited memory and computing, e.g., IoT devices [38]. Additionally, since real-time graph updating is a heavy operation in most cases, deploying large models even on cloud infra can lead to poor response times due to network latency. This is unacceptable for many real-time applications like our money laundering scenario [19]. One potential solution is employing real-time prediction in batches to reduce the overhead. Further, for a financial system, the simpler the AML model the better from an operational point of view. However, the simple model might not be able to capture all the useful signals and may not effectively capture true money laundering. We would argue on selecting the best model, which is efficient and light at the same time, yet yields both lower false positives and false negatives. There should be a balance between selecting the best model regarding its effectiveness for AML and its underlying deployment infrastructure.

Nevertheless, since pipeline methods involving GE and classification take considerably longer time than their end-to-end counterparts, our empirical study recommends that the end-to-end approach to AML may be more efficient in terms of accuracy and computation time. Based on these considerations, integrating an end-to-end AML model into a real financial system would be more convenient. Moreover, scalable model deployment and inferencing pipeline backed by GPU support ensure faster inferencing in a real-time setting.

V. CONCLUSION AND OUTLOOK

In this paper, we employ semi-supervised graph learning techniques on financial transaction graphs to detect potential money laundering activities, involving gather scattered, scatter gathered, and cycles. We trained Node2Vec, Attri2Vec, GraphSAGE, and DGT embedding models to embed graph nodes into a lower dimensional vector space. RF, XGBoost, and LightGBM classifiers were then trained on embedding space to predict the suspiciousness of a node being involved in money laundering. Besides, we trained SkipGCN, FastGCN, and EvolveGCN in the end-to-end setting for the same. Our findings on several datasets show that graph analytics can be

an effective means for identifying laundering transactions. The representation learning power of GE and GNN-based models applied to graphs tends to improve node classification accuracy. Nevertheless, the global and local factors that we identify can help highlight the predominant factors in money laundering cases and elucidate the mechanisms of illicit fund transfers between nodes to enhance the interpretability and transparency of an AML model.

Even though combining embeddings with local and global node features significantly boosted the performance of pipeline methods, end-to-end methods consistently outperformed all pipeline methods, albeit the XGBoost model benefited from temporal information captured by different embedding models. Therefore, an end-to-end approach to AML may not only be more efficient w.r.t accuracy and computation time but also integrating an end-to-end AML model into a financial system would be more convenient. On the other hand, since graph analytics to AML is relatively new research, leaving much room for improvements. `First`, there are many other techniques employed by criminals to hide their illicit funds. Therefore, it is crucial to have effective AML laws and regulations, as well as vigilant and cooperative authorities to combat this global problem. `Second`, our semi-supervised learning approach to publicly available data is less sensitive, hence there is no risk of exposing sensitive information. However, similar to deep models that are vulnerable to adversarial attacks, AML detection mechanisms are susceptible to attacks too, e.g., an adversary having access to a model could manipulate transactions to trick the AML detection system into misclassifying illicit transactions as licit [41].

`Third`, in real banking settings, money launderers may take advantage of disclosing private transactions across banks and splitting transferred amounts into intermediaries. Differential privacy [42] could be a potential approach. Some attacks can be prevented by not disclosing the data,[15] by using cryptographic primitives, and by making available the model to trusted authorities only. Nevertheless, research [4] has outlined the incorporation of effective differentially private graph topology and model-sharing techniques in distributed banking settings. Further, a functional encryption schema by leveraging similarity calculations is investigated [44] to provide the financial authorities access to data in a federated learning setting in banks. In such a setting, the vector size would be as large as the number of accounts in the bank. In contrast, using KGE methods that are compatible with new privacy and data accessibility constraints, the size could be reduced significantly.

In this regard, a variant of the fast random projection described below could be a potential idea to obtain transaction/edge embeddings:

1) Each node n in the subgraph and its direct neighbours receives an initial random embedding $e_{n,0}$ from a public hash function on the node, under the constraint that no output is null.
2) For each node n in the subgraph, denoting N the set of its direct neighbours, and $w_{i\to j}$ the amounts sent from account i to account j (i.e., the weight on oriented edge from i to j, for any nodes i, j, and setting that weight to 0 when the nodes are not connected), n receives a new embedding: $e_{n,1} := \sum_{m\in N} e_{m,0}\cdot w_{m\to n}$ (as for outgoing transaction embeddings: $e_{n,1} := \sum_{m\in N} e_{m,0}\cdot w_{n\to m}$).
3) Vectors from step 2 are then normalized; each node n in subgraph receives embedding: $e_{n,3} := \frac{e_{n,2}}{||e_{n,2}||_2}$, where $||.||_2$ is the euclidean norm and $e_{n,\text{in}}$ is the embedding (resp. $e_{n,\text{out}}$ for outgoing transactions).

In future, we intend to introduce sophisticated attacks on the model, followed by exploring defence strategies. This will enable us to perform both inner-products and similarity calculations on embedding vectors, e.g., $\sigma(n,m) = \langle e_{n,\text{in}}; e_{m,\text{in}}\rangle \times \langle e_{n,\text{out}}; e_{m,\text{out}}\rangle$ by comparing nodes n from bank 1 and m from bank 2. In such a setting, nodes showing high similarity values should be labelled as suspicious accounts which might be part of the same money-laundering network, having the bulk of their transactions from and to the same neighbours.

`Fourth`, with the rapid development of a cashless society engaged in global economic exchange, the advent of cryptocurrency has catalyzed a paradigm shift in peer-to-peer transactions and extranational financial governance. Cryptocurrencies not only impose great challenges to AML but also increase difficulty across cryptocurrency types. Another challenge is temporal dynamics with the emergence/disappearance of new entities in the blockchain. Weber et al. [5] have shown that at timestep the market may appear to follow *Dark Market shutdown*, where no models (including EvolveGCN or DGT) would be able to capture such high volatility and consequently may not perform well.

`Fifth`, real-life data is more noisy and full of uncertainties. Therefore, sophisticated methods need to be developed to capture complex laundering patterns with extremely low illicit ratios [2]. This could be the case for synthetic data too, where even efficient AML models may face challenges regarding true detection of money laundering in the presence of lower illicit ratios and longer periods of laundering patterns [2]. A concrete example outlined by Altman et al. [2] is the LI version of the IBM ML dataset. Thus, we intend to focus on such datasets as well, with a focus on node classification.

`Sixth`, it is hard to initialise the node features. Therefore, we would like to employ self-supervised pre-training for the sake of feature initialisation. We believe that above mentioned reactive and proactive measures would help improve both the representation learning capability and adversarial robustness of an AML model. Nevertheless, we hope the approach presented in this paper will make non-trivial contributions and give network security and financial crime analysis some insights into how to employ semi-supervised graph learning

[15] For example, what if attackers have access to the model to perform input reconstruction or membership inference [43].

on large-scale transaction graphs for effective identification of potential money laundering cases.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their useful comments and feedback that helped improve this manuscript.

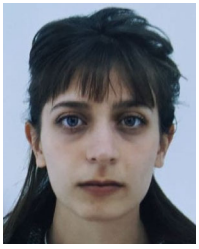
REFERENCES

[1] R. Frumerie, "Money laundering detection using tree boosting and graph learning algorithms," School Eng. Sci., KTH Roy. Inst. Technol., Stockholm, Sweden, Tech. Rep. diva2:1663255, 2021.
[2] E. Altman, J. Blanuša, L. von Niederhäusern, B. Egressy, A. Anghel, and K. Atasu, "Realistic synthetic financial transactions for anti-money laundering models," 2023, *arXiv:2306.16424*.
[3] *Money Laundering*. Accessed: Oct. 13, 2023. [Online]. Available: https://www.unodc.org/romena/en/money-laundering.html
[4] R. Soltani, U. T. Nguyen, Y. Yang, M. Faghani, A. Yagoub, and A. An, "A new algorithm for money laundering detection based on structural similarity," in *Proc. IEEE 7th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2016, pp. 1–7.
[5] M. Weber, J. Chen, T. Suzumura, A. Pareja, T. Ma, H. Kanezashi, T. Kaler, C. E. Leiserson, and T. B. Schardl, "Scalable graph learning for anti-money laundering: A first look," 2018, *arXiv:1812.00076*.
[6] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 38 3–7, 2018, Proceedings 15*. Springer, 2018, pp. 593–607.
[7] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "FlowScope: Spotting money laundering based on graphs," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4731–4738.
[8] Q. Rajput, N. S. Khan, A. Larik, and S. Haider, "Ontology based expert-system for suspicious transactions detection," *Comput. Inf. Sci.*, vol. 7, no. 1, p. 103, Jan. 2014.
[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[10] K. Michalak and J. Korczak, "Graph mining approach to suspicious transaction detection," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Sep. 2011, pp. 69–75.
[11] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "FRAUDAR: Bounding graph fraud in the face of camouflage," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 895–904.
[12] M. Cardoso, P. Saleiro, and P. Bizarro, "LaundroGraph: Self-supervised graph representation learning for anti-money laundering," in *Proc. 3rd ACM Int. Conf. AI Finance*, Nov. 2022, pp. 130–138.
[13] H. Kanezashi, T. Suzumura, X. Liu, and T. Hirofuchi, "Ethereum fraud detection with heterogeneous graph neural networks," 2022, *arXiv:2203.12363*.
[14] S. Xi Rao, S. Zhang, Z. Han, Z. Zhang, W. Min, Z. Chen, Y. Shan, Y. Zhao, and C. Zhang, "XFraud: Explainable fraud transaction detection," 2020, *arXiv:2011.12193*.
[15] Z. Chen, L. Chen, S. Villar, and J. Bruna, "Can graph neural networks count substructures?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 10383–10395.
[16] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 657–668, Jan. 2023.
[17] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeuralIPS*, vol. 30, 2017, pp. 1025–1035.
[18] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," 2018, *arXiv:1801.10247*.
[19] M. R. Karim, T. Islam, M. Shajalal, O. Beyan, C. Lange, M. Cochez, D. Rebholz-Schuhmann, and S. Decker, "Explainable AI for bioinformatics: Methods, tools and applications," *Briefings Bioinf.*, vol. 24, no. 5, Sep. 2023, Art. no. bbad236.
[20] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, and S. Staab, "Knowledge graphs," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–37, 2021.
[21] Y. Dai, S. Wang, N. N. Xiong, and W. Guo, "A survey on knowledge graph embedding: Approaches, applications and benchmarks," *Electronics*, vol. 9, no. 5, p. 750, May 2020.
[22] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu, "Knowledge graph embedding by flexible translation," in *Proc. 15th Int. Conf. Principles Knowl. Represent. Reasoning*, 2016, pp. 557–560.
[23] M. R. Karim, M. Cochez, J. B. Jares, M. Uddin, O. Beyan, and S. Decker, "Drug-drug interaction prediction based on knowledge graph embeddings and convolutional-LSTM network," in *Proc. 10th ACM Int. Conf. Bioinf., Comput. Biol. Health Informat.*, Sep. 2019, pp. 113–123.
[24] P. Xia, Z. Ni, H. Xiao, X. Zhu, and P. Peng, "A novel spatiotemporal prediction approach based on graph convolution neural networks and LSTM for money laundering fraud," *Arabian J. Sci. Eng.*, vol. 47, no. 2, pp. 1–17, 2021.
[25] Z. Al Sahili and M. Awad, "Spatio-temporal graph neural networks: A survey," 2023, *arXiv:2301.10569*.
[26] G. Jin, Y. Liang, Y. Fang, Z. Shao, J. Huang, J. Zhang, and Y. Zheng, "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 1, no. 1, pp. 1–20, Jan. 2024, doi: 10.1109/tkde.2023.3333824.
[27] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5363–5370.
[28] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, Q. Chen, and V. C. Lee, "Anomaly detection in dynamic graphs via transformer," *IEEE Trans. Knowl. Data Eng.*, vol. 1, no. 1, pp. 1–14, Dec. 2023, doi: 10.1109/tkde.2021.3124061.
[29] M. Cochez, P. Ristoski, S. P. Ponzetto, and H. Paulheim, "Biased graph walks for RDF graph embeddings," in *Proc. 7th Int. Conf. Web Intell., Mining Semantics*, Jun. 2017, pp. 1–12.
[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[31] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Attributed network embedding via subspace discovery," *Data Mining Knowl. Discovery*, vol. 33, no. 6, pp. 1953–1980, Nov. 2019.
[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, vol. 3, 3rd ed. Cambridge, MA, USA: MIT Press, 2009, sec. 6, p. 7.
[33] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, nos. 1–7, pp. 107–117, Apr. 1998.
[34] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *Proc. 20th Int. Conf. World Wide Web*, Mar. 2011, pp. 607–614.
[35] F. Di Castro and E. Bertini, "Surrogate decision tree visualization," in *Proc. IUI Workshops*, 2019, pp. 1–5.
[36] R. I. T. Jensen, J. Ferwerda, K. S. Jørgensen, E. R. Jensen, M. Borg, M. P. Krogh, J. B. Jensen, and A. Iosifidis, "A synthetic data set to benchmark anti-money laundering methods," *Sci. Data*, vol. 10, no. 1, p. 661, Sep. 2023.
[37] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics," 2019, *arXiv:1908.02591*.
[38] M. R. Karim, M. Shajalal, A. Graß, T. Döhmen, S. A. Chala, A. Boden, C. Beecks, and S. Decker, "Interpreting black-box machine learning models for high dimensional datasets," in *Proc. IEEE 10th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2023, pp. 1–10.
[39] C. Molnar, *Interpretable Machine Learning*. Lulu.com, 2020.
[40] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Proc. NeuralIPS*, 2019, pp. 9240–9251.
[41] D. Berman, A. Buczak, J. Chavis, and C. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, Apr. 2019.
[42] M. Abadi, I. Goodfellow, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.
[43] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 1–37, Jan. 2022.
[44] P. de Perthuis and D. Pointcheval, "Two-client inner-product functional encryption with an application to money-laundering detection," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 725–737.

MD. REZAUL KARIM received the B.Sc. degree from the University of Dhaka, Bangladesh, the M.Sc. degree from Kyung Hee University, South Korea, and the Ph.D. degree from RWTH Aachen University, Germany. He was a Lead Software Engineer with Samsung Electronics, South Korea. He was also an ML Engineer with the Insight Centre for Data Analytics, University of Galway, Ireland. He is currently a Staff Data Scientist with ALDI SÜD-International Data and Analytics, Germany, and a Visiting Researcher with RWTH Aachen University. Before joining ALDI SÜD, he was a Senior Data Scientist with Fraunhofer FIT and a Postdoctoral Researcher with RWTH Aachen University. His research interests include applied machine learning, explainable AI (XAI), generative AI, and knowledge graphs.

FELIX HERMSEN received the M.Sc. degree in computer science from RWTH Aachen University, Germany, in 2021. Since 2022, he has been a Researcher with the Fraunhofer Institute for Applied Information Technology FIT, Germany, where he is currently with the Data Protection and Sovereignty Research Group. His research interests include privacy-preserving machine learning and next-generation marketplaces.

SISAY ADUGNA CHALA received the Ph.D. degree in computer science from the University of Siegen, Germany. He was a Marie Curie ITN Fellow working on unsupervised feature learning from textual data applied in data-intensive personalized vacancy recommendation with the University of Siegen. He involved in research on statistical machine translation with German Research Center for Artificial Intelligence (DFKI). He is currently a Postdoctoral Researcher with Fraunhofer FIT and the RWTH Aachen University, Germany, and the Deputy Head of the Intelligent Data Analytics Group, Department of Data Science and AI, Fraunhofer FIT. His research interests include machine translation, data mining and machine learning, information retrieval, and the application of AI in various problem domains.



PAOLA DE PERTHUIS is currently pursuing the Ph.D. degree in cryptography with École Normale Supérieure (ENS), Paris, and Cosmian, France. Previously, she worked on cryptographic systems for the detection of money laundering networks, to which she intends to add simplified embedding techniques compatible with their security models.

AVIKARSHA MANDAL received the M.Sc. degree in information and computer sciences from the University of Luxembourg, Luxembourg, in 2012, and the Ph.D. degree in applied cryptography from the University of Mannheim, Germany, in 2020. From 2013 to 2019, he was a Research Assistant in IT security with the Offenburg University of Applied Sciences, Germany. Since 2019, he has been a Senior Researcher with the Fraunhofer Institute for Applied Information Technology FIT, Germany, where he is currently the Head of the Data Protection and Sovereignty Research Group. His research interests include improving data privacy and security in data-driven applications across different domains, such as cybersecurity, energy, and blockchain.