DELATOR: Money Laundering Detection via Multi-Task Learning on Large Transaction Graphs

Henrique S. Assumpção
Universidade Federal de Minas Gerais
henriquesoares@dcc.ufmg.br

Fabrício Souza
Universidade Federal de Minas Gerais
fabricio.souza@dcc.ufmg.br

Leandro Lacerda Campos
InterMind (Inter&Co)
Universidade Federal de Minas Gerais
leandro.campos@inter.co

Vinícius T. de Castro Pires
InterMind (Inter&Co)
Universidade Federal de Minas Gerais
vinicius.pires@inter.co

Paulo M. Laurentys de Almeida
InterMind (Inter&Co)
paulo.laurentys@inter.co

Fabrício Murai
Worcester Polytechnic Institute
Universidade Federal de Minas Gerais
fmurai@wpi.edu

Abstract—Money laundering has become one of the most relevant criminal activities in modern societies, as it causes massive financial losses for governments, banks and other institutions. Detecting such activities is among the top priorities when it comes to financial analysis, but current approaches are often costly and labor intensive partly due to the sheer amount of data to be analyzed. Hence, there is a growing need for automatic anti-money laundering systems to assist experts. In this work, we propose DELATOR, a novel framework for detecting money laundering activities based on graph neural networks that learn from large-scale temporal graphs. DELATOR provides an effective and efficient methodology from an easily imbalanced graph data, by adapting concepts from the GraphSMOTE framework and incorporating elements of multi-task learning to obtain rich node embeddings for node classification. DELATOR outperforms all considered baselines, including an off-the-shelf solution from Amazon AWS by 23% with respect to AUC-ROC. We also conducted real experiments that led to the discovery of 7 new suspicious cases among the 50 analyzed ones, which have been reported to the authorities.

Index Terms—Money Laundering Detection, Graph Neural Networks, Multi-Task Learning, Large Transaction Graphs

I. INTRODUCTION

Money laundering is a general term referring to a myriad of different criminal activities that seek to legitimize illicit financial gains or to conceal the origin of money transferred to illegal entities, e.g., terror financing and drug trafficking [1]. Although varied in nature, money laundering activities can be categorized into three well-defined stages that represent the full process of legitimizing a series of illegal economic activities: (i) placement, (ii) layering, and (iii) integration [2]. Placement refers to the initial process of introducing the illicitly gained assets into the legitimate financial system, by removing obvious traces of illegality. Layering is one of the most important and complex stages of money laundering. It consists of a series of transactions with no purpose other than to conceal the illicit origin of the money. Finally, integration aims to integrate these assets into the legal economy.

Most western countries implement a rule-based regulation that all banks must follow in order to comply with legislation [1], [3]. Such rules vary in nature, but can often describe numerical thresholds for flagging suspicious transactions, e.g., receiving or sending an amount that surpasses n times the client’s income. Traditionally, flagged clients are manually analyzed by an anti-money laundering (AML) team of experts, who try to determine the existence of extraordinary circumstances that could justify the threshold violation (e.g., a client sold a property they owned). Based on that analysis and the current legislation, the team decides whether to report the client to the authorities or not; however, this manual inspection of flagged clients is an expensive and time-consuming process. Not surprisingly, in Brazil, banks have 45 days to file reports regarding the transactions performed in a given month.

Such rule-based AML systems have three main limitations. First, it is hard to prioritize flagged clients, except by ranking them according to the number of rule violations1. A good prioritization scheme could significantly expedite the reporting of criminals and could provide side information to help experts close cases unlikely to be reported. Second, there is potentially a large number of clients “just below” thresholds for being flagged that will never be analyzed. Despite not violating any given rule, altogether, the set of transactions made by a client could provide evidence of a crime. Potential losses for a false negative are immense since the financial institution could face significant legal backlash. Last, the fact that some common transaction patterns (e.g., Smurfing [4]) occur in money laundering activities underlines the limitations of rules that apply to clients individually, namely, they do not account for structural information in the network formed by all transactions.

Therefore, there is a notable demand for new data-driven tools for money laundering detection that escape the rule-based approaches. In this paper, we will focus on building a robust data-driven framework for detecting money laundering activity in the layering stage by adapting different machine learning models and novel approaches for graph-structured

1 Although a violation severity score could be computed for each rule, it is not obvious how to combine them into a single score.
data. We propose a multi-task learning framework named DELATOR for detecting money laundering in dynamic financial transaction networks. For each graph snapshot, it generates client (node) embeddings based both on the unsupervised link prediction task and on a supervised edge regression task where the labels are the transaction values. Next, the embeddings of each client are concatenated over the snapshots. DELATOR then generates synthetic suspicious clients in the embedding space to oversample the minority class. Last, it trains a supervised classifier for predicting the probability of a given client being involved in money laundering.

Thus, we summarize our contributions as follows:

- We propose a scalable and effective AML framework that experimentally outperforms state-of-the-art methods and has a relatively simple implementation.
- DELATOR is among the first methods for detecting money laundering activity on large transaction graphs that are temporal, heterogeneous, and have extremely imbalanced target classes. By leveraging different aspects of the network, we are able to simplify the data modeling and create a method that performs well on a real-world large dataset.
- We evaluate our framework in a real setting by performing data-driven inference on millions of bank accounts, which ultimately led to the detection of multiple suspicious clients, that were then reported to the authorities for potential involvement in money laundering activities. To the best of our knowledge, this is the first system of its kind to be employed successfully in the context of Brazilian banks.

II. RELATED WORK

A. Graph Neural Networks

Our work is mostly related to graph representation learning, and more specifically, learning node representations on a latent space based on unsupervised or semi-supervised tasks. In recent years, models based on Graph Neural Networks (GNNs) [5] have taken a prominent role in the context of learning latent representations for graph-structured data. GraphSAGE [6] was among the first unsupervised learning frameworks that could yield an inductive model for graphs. Its loss function is based on random walks, and it encourages nearby nodes to have similar embeddings, while ensuring that distant nodes have dissimilar embeddings. The Graph Attention architecture [7] has also been successful in many different graph-related tasks, and it utilizes masked self-attention during the message passing mechanism, allowing the network to learn which neighbors to prioritize when aggregating information.

B. Class Imbalance

Class imbalance refers to a significant difference in the number of instances of each class and is a characteristic displayed by many real-world datasets. It tends to bias model predictions towards the majority classes.

There are many strategies for dealing with this problem, some of which seek to adjust class size through over- or undersampling, i.e., making the majority classes smaller and the minority classes bigger. SMOTE [8] is one of the most popular oversampling algorithms, and it addresses the problem by interpolating samples belonging to minority classes with their nearest neighbors in order to create new synthetic samples.

GraphSMOTE [9] adapts the SMOTE algorithm to better suit graph representation learning applications. This method first extracts node embeddings from the graph by using a single-layer GNN, and then applies SMOTE to each minority class on the latent embedding space, thus ensuring that the oversampling process will leverage the topological information encoded by the embeddings. After an extensive analysis of the available literature, we decided to adapt some of the ideas from GraphSMOTE – which was originally designed for static graphs – to our framework in order to improve the process of learning information about our dynamic network.

C. Money Laundering Detection

Detecting money laundering activity on financial transaction networks can be seen as a particular case of fraud detection on graph-based data. In this more general context, most of the problems are related to imbalanced scenarios, where a small minority of individuals are actually involved in the target illicit activity.

The authors of [10] provide a simple and efficient method for detecting suspicious activity specifically related to money laundering, by employing a series of standard database join operations to detect sub-graphs that follow the Smurf patterns. The authors of [11] provide a GNN-based approach for money laundering detection on graphs, by employing an LSTM network coupled with a graph convolutional network in order to simultaneously model the topological and temporal relationships between nodes on the graph. However, the aforementioned models do not directly deal with the imbalance problem that is prominent in our data.

III. DATA MODELING

In this section, we describe the model considered in this work. A potential, more complex alternative is discussed in our tech report [12].

The available transaction data consists of a multiset \(\mathcal{E}\) of monetary transactions with partition \(\mathcal{E} = \bigcup_{t=1}^{T} \mathcal{E}^t\), where each \(\mathcal{E}^t\) represents all transactions made at timestep \(t \in \{1, 2, ..., T\}\), a set \(\mathcal{V}\) of users and a set \(\mathcal{X} = \{(x_1, y_1), ..., (x_n, y_n)\}\), where each \(x_i \in \mathbb{R}^d\) represents a given user’s attributes. Each \(y_i\) represents the target variable, indicating if the user was suspected of being involved in money laundering activity. Each element \(e = ((v, u), w, c) \in \mathcal{E}^t\) represents a transaction from user \(v\) to user \(u\), of amount \(w\) and type \(c \in \{1, 2, ..., m\}\), executed at timestep \(t\), and there is no restriction on the number of transactions between users. Since the data is dynamic in nature, it is natural to model it as a dynamic graph, more specifically as a Discrete-Time Dynamic Graph (DTDG), which is a sequence of static graph snapshots.
taken over certain time intervals. In our context, each snapshot contains the same set of nodes and node attributes.

We model the data as a homogeneous directed weighted DTDG, i.e., a sequence $S_{homo} = \{G^1, G^2, ..., G^T\}$ of graphs, where each $G^t \in S_{homo}$ represents the network at time $t$. In order to obtain a simple homogeneous graph at each snapshot $t$, we create a new edge set $E^t = \{((v, u), w') | w' = \sum_{e \in E^t : e_i = (v, u)} w \}$ where $e_i$ represents the $i$-th element of the tuple $e$ –, that aggregates all edges between each pair of nodes $(v, u)$ by setting $w'$ as the total amount transacted from $v$ to $u$. We can then define each snapshot as $G^t = (\mathcal{V}, E^t, \mathcal{X})$. This modeling aggregates transactions with different types into a single transaction, thus simplifying the graph structure when compared to heterogeneous approaches.

IV. PROPOSED FRAMEWORK

In this section, we provide a brief overview of DELATOR, and then proceed to detail each step of the framework. DELATOR is comprised of three main steps:

1) This step consists of a multi-task learning algorithm to obtain node embeddings for each graph snapshot. We first train a GNN model that aims to optimize the link prediction loss, i.e., a loss function that enforces embedding similarity between connected nodes, imbued with negative sampling. Next, we train a second GNN model that aims to optimize the edge regression loss, i.e., the task of predicting edge weights. The second network also generates node embeddings, that are then concatenated with the embeddings generated by the first GNN and passed through a Multi-Layered Perceptron (MLP) to obtain the final prediction for the edge weight. These networks are trained individually on their respective tasks and each generates node embeddings for all graph snapshots. We concatenate such embeddings into a single representation per snapshot, and then concatenate the representations across all snapshots in order to obtain a single time-aware description of the nodes in euclidean space;

2) Next, we create synthetic nodes to oversample the minority class, by applying the SMOTE algorithm directly to the training set in order to obtain new embeddings for the minority class.

3) We finally proceed to train a classifier on the aggregated oversampled data that predicts the probability of a user being suspected of being involved in money laundering.

A. Node Embedding Generation

The first part of the framework consists of extracting node embeddings from the network. We adopt a multi-task learning approach and thus subdivide the training procedure into two steps.

1) Link Prediction (LP): First, we train a GNN network to obtain node embeddings based on an unsupervised loss for link prediction, i.e., predicting if two nodes are connected in the graph. Our framework allows for any GNN architecture to be used in this step, and we will present an example using the GraphSAGE architecture. Recall the sequence $S_{homo}$ defined in Section III. For each graph snapshot $G^t \in S_{homo}$, the following equation describes the message passing mechanism for an arbitrary layer $l$ out of a total of $L$ layers of GraphSAGE:

$$h^t_{v,u} = f(W_l \cdot \text{AGG}(\{h^{t-1}_{v,u} \forall u \in \mathcal{N}(v)\}) || B_l \cdot h^{t-1}_{v,u})) \quad (1)$$

In (1), $h^t_{v,u}$ is the embedding of node $v$ at layer $l$ and snapshot $t$. $W_l$ is a set of learnable weights, $B_l$ is a set of learnable biases, $\|$ is the concatenation operator along the columns, AGG is an aggregation function, $\mathcal{N}(v)$ is the set of outgoing neighbours of node $v$, $f$ is an activation function, e.g., ReLU, and $\cdot$ denotes matrix multiplication. We initialize $h^0_{v,u} = x_v$ for all nodes. For simplicity, we will refer to $h^t_{v,u}$ as $h^t_v$. We can then define the link prediction loss $L^t_{lp}$ for a given snapshot $t$, that we wish to minimize, as follows:

$$L^t_{pos}(G^t) = \sum_{((v,u),w) \in E^t} \log(\sigma(h^t_v \cdot h^t_u)) \quad (2)$$

$$L^t_{neg}(G^t) = \sum_{((v,u),w) \in E^t} \log(1 - \sigma(h^t_v \cdot h^t_u)) \quad (3)$$

$$L^t_{lp} = \frac{L^t_{neg}(G^t) + L^t_{pos}(G^t)}{|E^t| + |E^t|} \quad (4)$$

In (2)-(4), $\sigma$ is the sigmoid function, $L^t_{pos}(G^t)$ denotes the log-likelihood of the link prediction for the current snapshot and $L^t_{neg}(G^t)$ denotes the log-likelihood of the link prediction for a negative sample of the current snapshot, i.e., we define a set $E^t$ of randomly selected edges, and $\top$ denotes the transpose operator. This loss function is a direct adaptation of the GraphSAGE unsupervised loss, and it encourages the model to create embeddings that are similar for connected nodes while enforcing a distinct representation for disconnected nodes. The GNN model is trained consecutively on $S_{homo}$, i.e., at each epoch, we compute the embeddings and scores according to the aforementioned equations, and then perform one optimization step at each snapshot.
2) **Edge Regression (ER):** After training a GNN model for link prediction, we now train a model to perform edge regression, i.e., predict the value of a given transaction between two users. The model consists of a separate GNN that generates embeddings $z^t_v$ for all nodes, and we again note that this step can be performed with any GNN architecture. For an arbitrary edge $e = ((v, u), w) \in E^t$, we obtain the predicted edge weight $\hat{w}(e)$ as follows:

$$\hat{w}(e) = \text{MLP}(h^t_v \parallel z^t_v \parallel h^t_u \parallel z^t_u)$$  \hspace{1cm} (5)

In (5), the predicted edge weight is obtained by concatenating the link prediction embeddings (fixed) and edge regression embeddings (trainable) of both $v$ and $u$, and then passing them through a MLP network. We can then define the edge regression loss $L^t_{er}$ for a given snapshot $t$, that we wish to minimize, as follows:

$$l^t_{er}(w, \hat{w}) = \begin{cases} 
\frac{(\hat{w} - w)^2}{2\gamma^2}, & \text{if } |\hat{w} - w| < \gamma \\
|\hat{w} - w| - 0.5\gamma, & \text{otherwise,}
\end{cases} \hspace{1cm} (6)

L^t_{er} = \frac{1}{|E^t|} \sum_{e=((v,u),w)\in E^t} l^t_{er}(w, \hat{w}(e)).$$

In (6), $\gamma$ denotes a threshold hyperparameter, and $l^t_{er}(w, \hat{w})$ denotes the partial loss for a given edge. Equation (6) represents the smooth L1 loss function of the edge regression task. We also train this model on $S_{homo}$ in a consecutive fashion. After training the models, we obtain a single time-aware embedding for a given node $v$, denoted by $\eta(v)$, as follows:

$$\eta(v) = \frac{1}{T} \sum_{t=1}^{T} (h^t_v \parallel z^t_v).$$  \hspace{1cm} (7)

### B. Oversampling

We now seek to generate synthetic nodes to oversample the minority class. In this work we decided to adopt the SMOTE algorithm, however our framework is compatible with any oversampling methods that can use the generated embeddings.

The intuition behind SMOTE is to perform interpolation on samples from the minority class with their nearest neighbors in the embedding space of the same class, in order to generate new embeddings that are similar to the ones found in this minority class. Given a node $v \in V$ belonging to a certain minority class, e.g., individuals involved in money laundering, consider the final embedding $\eta(v)$. SMOTE generates a new sample $v'$ according to the following equation:

$$\eta(v') = (1 - \lambda) \cdot \eta(v) + \lambda \cdot \eta(\text{nn}(v))$$  \hspace{1cm} (8)

In (8), $\text{nn}(v)$ denotes the nearest neighbor of $v$ from the same class, according to the euclidean distance in the embedding space, and $\lambda \sim U(0, 1)$ denotes a uniform random variable. Since the new node is generated via interpolation of nodes from the same class, we can label the new synthetic node $v'$ as belonging to the same class as $v$, thus we can artificially oversample the minority class to overcome the imbalanced scenario. We highlight that this oversampling step is only performed on the training data.

### C. Node Classification

The last step of DELATOR consists of a supervised classification task for detecting users suspected of being involved in money laundering activity. After oversampling the minority class, we employ LightGBM [13], a state-of-the-art gradient boosting algorithm for supervised learning. The LightGBM model takes the oversampled node embeddings as input and outputs a probability classification of the users, i.e., a real value between 0 and 1 indicating the probability of a given user being suspicious. We highlight that our framework supports any supervised classifier at this step.

### V. EXPERIMENTS

In this section, we detail the experiments conducted to evaluate the performance of DELATOR. Specifically, we are interested in the following questions:

1) Can DELATOR’s approach perform well on complex, heterogeneous relational networks?

2) How well does DELATOR perform when compared to other state-of-the-art techniques for node embeddings and node classification?

3) Can DELATOR help the AML team detect suspicious users on a real-world experiment?

For further details regarding the experiments discussed in this section, please refer to our technical report [12].

### A. Experimental Setup

1) **Dataset:** We use the dataset from Inter&Co, one of the biggest and fastest growing digital banks in Brazil. The dataset is composed of transactions made to/from clients of the bank.

The dataset is comprised of a sample of 20 million users, 200 transaction types, and 110 million transactions that span over a month. We split this data into 5 distinct snapshots, each approximately representing a week. As mentioned before, all snapshots contain the same set of users, however, the number of transactions varies between them. Many of the users in the network are not clients from Inter – referred to as non-Inter clients –, e.g., clients from other banks or financial institutions. Classifying such non-Inter clients is beyond the scope of this work, and thus we only keep them during the node embedding generation step (Step 1 in Section IV) due to their topological value for understanding the overall structure of the network, after which they are no longer considered.

The minority class (‘suspicious’) of the dataset represents if the Inter client was suspected of being involved in money laundering, and thus reported to the authorities by Inter’s AML team. The majority class (‘non-suspicious’) represents Inter clients that: (i) did not trigger any of the rules mentioned in Section I, or (ii) triggered a number of rules and, after manual analysis by the AML team, were discarded as a suspect. The task in question is to detect Inter clients that were suspected of being involved in money laundering activity, and the imbalance ratio between the classes – the ratio between the number of Inter clients in the minority class and the majority class – is $2 \cdot 10^{-5}$. 

712
2) Baselines: We now detail the state-of-the-art methods used as baselines. First, we consider GNN baseline methods for extracting node embeddings from graphs, often referred to as embedding generators:

- **DGL-KE** [14]: A state-of-the-art framework developed by Amazon AWS for learning representations on knowledge graphs, i.e., heterogeneous graphs with multiple edge and node types.
- **GraphSAGE (SAGE)**: A state-of-the-art GNN architecture that generalizes the original model for learning on graphs, by allowing for multiple different aggregator functions on the message passing step.
- **Graph Attention (GAT)**: A state-of-the-art GNN architecture that introduces an attention mechanism to the message passing algorithm.
- **Graph Convolution (GCN)**: The first proposed GNN architecture based on message passing on graphs.

We train the models on an unsupervised link prediction task to obtain node embeddings, then we oversample the minority class on the embedding space for the training set, and last, we train a supervised classifier in the final classification task. We employ the modeling described in Section III to train the SAGE, GAT, and GCN methods. DGL-KE utilizes a heterogeneous modeling to obtain node embeddings, described in detail in [14]. We also experimented with different architectures of embedding generators for DELATOR for the first step of the framework, i.e., obtaining embeddings based on both the link prediction and edge regression tasks. We identify these variants by their names on the subscript, e.g., DELATOR\_SAGE denotes the version utilizing GraphSAGE layers on both tasks.

As a baseline that does not utilize the network’s transaction information in its modeling, we consider CatBoost [15], a state-of-the-art gradient boosting algorithm for supervised learning, that allows for better performance on highly categorical feature spaces. This model learns directly on the client attributes $X$ in order to obtain the final prediction. Since the raw feature space is mostly categorical, the CatBoost model does not oversample the minority class.

3) Evaluation Metrics: For evaluating the effectiveness of the proposed framework for detecting suspicious clients in transaction networks, we propose the following metrics:

- **AUC-ROC**: area under the ROC curve (True positive rate vs. False positive rate).
- **AUPR**: area under the PR curve (Precision vs. Recall).
- **F1-Fraud**: geometric mean between precision and recall w.r.t. the label that represents suspicious activity.
- **Maximum F1-Fraud**: maximum value of the F1-Fraud w.r.t. all possible thresholds.

B. Results

1) Quantitative Results: To answer the first two questions posed in this section, we compared the results of DELATOR with the proposed baselines for the final task of detecting suspicious clients.

Table I shows the overall evaluation results for the final prediction task. We observe that, for all metrics, the versions of DELATOR have outperformed the proposed baselines, especially when compared to DGL-KE. The results w.r.t. AUC-ROC show that, on average, DELATOR\_SAGE provides a more consistent ranking of clients when compared to the other baselines. The AUPR score is also higher for DELATOR\_GCN, showing that it is easier to select a classification threshold such that the framework’s precision dominates its recall. In addition, we note that the F1-Fraud and Maximum F1-Fraud metrics are also higher for both DELATOR\_SAGE and DELATOR\_GCN. This is a strong indicator that the framework has the best performance for correctly classifying clients that are suspected of being involved in money laundering.

Catboost presents a higher standard deviation for the AUPR metric when compared DELATOR\_SAGE, coupled with lower mean values for the AUC-ROC, F1-Fraud and Maximum F1-Fraud metrics when compared to all other methods, indicating that CatBoost struggles to perform consistently on our dataset.

Fig. 2 provides the ROC and PR curves for the best versions of DELATOR, GCN, and DGL-KE (based on validation sets). We observe that the ROC curve for DELATOR increases in height at a much faster pace than the baselines, signaling that it is easier to choose a classification threshold for DELATOR such that the true positive rate is high while the false positive rate is reasonably low, i.e., DELATOR is better at detecting suspicious clients. The best version of DELATOR also has a considerably higher AUC-ROC when compared to DGL-KE and GCN. The PR curve for DELATOR shows a much more desirable behavior, as it does not decrease as fast as DGL-KE’s or GCN’s.

![Fig. 2: ROC and PR curves for the best versions of DELATOR, GCN and DGL-KE, based on validation sets, for the final task of detecting suspicious clients](image-url)

2) Real-World Experiment: In order to answer the final question posed in this section, we also conducted a real-world experiment with Inter to verify the performance of DELATOR. As previously discussed, most Brazilian banks implement a flawed rule-based system for detecting clients suspected of being involved in money laundering that is extremely ineffective. DELATOR can provide significant help for the AML team due to its low training and prediction time, allowing it to produce a list of most likely suspicious clients effectively. In this context, we perform an experiment consisting of ranking all clients that did not trigger any alarms during a given

713
TABLE I: Evaluation results for the final prediction task of detecting suspicious clients.

| Method        | Evaluation Metrics | | | |
|---------------|--------------------|---|---|---|
|               | AUC-ROC | AUPR | F1-Fraud | Max. F1-Fraud |
| CatBoost      | 0.653 ± 0.075 | 0.007 ± 0.015 | 0 ± 0 | 0.009 ± 0.003 |
| DGL-KE        | 0.755 ± 0.028 | 0 ± 0 | 0 ± 0 | 0.001 ± 0 |
| SAGE          | 0.846 ± 0.039 | 0.001 ± 0 | 0.000 ± 0 | 0.019 ± 0.017 |
| GAT           | 0.743 ± 0.043 | 0.001 ± 0.001 | 0.001 ± 0.001 | 0.025 ± 0.019 |
| GCN           | 0.870 ± 0.018 | 0.002 ± 0.002 | 0.002 ± 0 | 0.032 ± 0.025 |
| DELATOR (GCN) (Ours) | 0.889 ± 0.014 | **0.011 ± 0.001** | 0.003 ± 0.001 | **0.049 ± 0.043** |
| DELATOR (GAT) (Ours) | 0.882 ± 0.023 | 0.001 ± 0 | 0.003 ± 0.001 | 0.018 ± 0.015 |

time interval. We consider the top 50 clients with highest probability of being classified as ‘suspicious’ according to DELATOR. These clients were manually analyzed by the AML team, which resulted in the discovery of 7 new clients involved in suspicious activity, that were then reported to the authorities. This experiment shows that DELATOR can create a new effective and efficient way of detecting suspicious clients that would be otherwise mislabeled as ‘non-suspicious’ by the rule-based system, as well as providing a way of detecting possible rule failures which can then be updated by the analysts, thus vastly enhancing the capabilities of the AML team.

VI. CONCLUSION AND FUTURE WORK

In this work, we introduce a novel framework for detecting money laundering activity in large transaction graphs. By leveraging different information available in the network, we are able to construct a multi-task learning procedure that generates time-aware node representations on euclidean space, which then allows us to perform a supervised learning procedure in order to predict the probability of a given client being involved in money laundering. This procedure is robust to highly imbalanced target classes, allowing for an effective method for learning on dynamic networks that suffer from large class imbalance. We then perform a series of experiments to evaluate the performance of our model, and experimentally demonstrate that DELATOR outperforms all considered baselines w.r.t. the evaluation metrics. We also perform a real-world experiment with the help of Inter’s AML team to evaluate DELATOR’s performance in a practical scenario, and show that the model was able to detect multiple clients involved in suspicious activity. Overall, we have shown that DELATOR provides an efficient and effective way of detecting money laundering activity and that the framework can provide significant help to banks and financial institutions.

In the future, we plan to test DELATOR in other datasets related to money laundering detection. We also intend to test the generalization capabilities of the framework in other tasks related to financial activities, such as financial fraud detection.

REFERENCES

[1] FATF, “International standards on combating money laundering and the financing of terrorism & proliferation.” 2012-2021.

[2] E. Ebikake, “Money laundering: An assessment of soft law as a technique for repressive and preventive anti-money laundering control,” Journal of Money Laundering Control, vol. 19, no. 4, pp. 346–375, 2016.

[3] BACEN, “Circular nº 3.978, de 23 de janeiro de 2020,” 2020. [Online]. Available: https://www.in.gov.br/web/dou/-/circular-n-3.978-de-23-janeiro-de-2020-20236631175

[4] S. N. Welling, “Smurfs, money laundering and the federal criminal law: The crime of structuring transactions,” vol. 41, pp. 287–343, 1989.

[5] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” CoRR, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[6] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” CoRR, abs/1706.02216, 2017. [Online]. Available: http://arxiv.org/abs/1706.02216

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” 2018.

[8] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” CoRR, vol. abs/1106.1813, 2011. [Online]. Available: http://arxiv.org/abs/1106.1813

[9] T. Zhao, X. Zhang, and S. Wang, “Graphsmote: Imbalanced node classification on graphs with graph neural networks,” in Proceedings of the 14th ACM international conference on web search and data mining, 2021, pp. 833–841.

[10] M. Starnini, C. E. Tsourakakis, M. Zanamanpour, A. Panisson, W. Allasia, M. Fornasier, L. L. Puma, V. Ricci, S. Ronchini, A. Ugrinoska, M. Varetto, and D. Moncalvo, “Smurf-based anti-money laundering in time-evolving transaction networks,” in Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track. Springer International Publishing, 2021, pp. 171–186. [Online]. Available: https://doi.org/10.1007%2F978-3-030-86514-6_11

[11] I. Alarab and S. Prakoonwit, “Graph-based lstm for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data,” Neural Processing Letters, Jun 2022. [Online]. Available: http://arxiv.org/abs/10.1007/s11063-022-10904-8

[12] H. S. Assumpção, F. Souza, L. L. Campos, V. T. d. C. Pires, P. M. L. de Almeida, and F. Murai, “Delator: Money laundering detection via multi-task learning on large transaction graphs,” 2022. [Online]. Available: https://arxiv.org/abs/2205.10293

[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “Lightgbm: A highly efficient gradient boosting decision tree,” in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde8486e9bbddeb0b746-Paper.pdf

[14] D. Zheng, X. Song, C. Ma, Z. Tan, Z. Ye, J. Dong, H. Xiong, Z. Zhang, and G. Karypis, “Dgl-ke: Training knowledge graph embeddings at scale,” in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 739–748.

[15] A. V. Gorogush, A. Gulin, G. Gusev, K. N. Kazeev, L. O. Prokhorenkova, and A. Vorobev, “Fighting biases with dynamic boosting,” CoRR, vol. abs/1706.09516, 2017. [Online]. Available: http://arxiv.org/abs/1706.09516

[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “Lightgbm: A highly efficient gradient boosting decision tree,” in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde8486e9bbddeb0b746-Paper.pdf