Combating Camouflage and Forgetting: Spatio-Temporal Dual Denoising for Money Laundering Detection

Xueting Yang
Tongji University
yangxt@tongji.edu.cn

Zhong Li *
Donghua University
007lizhong@dhu.edu.cn

Changjun Jiang
Tongji University
cjjiang@tongji.edu.cn

Wenbin Zhang
Florida International University
wenbin.zhang@fiu.edu

Meikang Qiu
Augusta University
qiumeikang@yahoo.com

Abstract

Money laundering detection on financial transaction graphs is critical but challenging. Laundering activities often conceal illicit fund flows within dense legitimate transaction neighborhoods and disperse suspicious behaviors over extended time spans. These patterns cause weak illicit signals to be spatially diluted and temporally forgotten, limiting the effectiveness of existing graph learning methods in anti-money laundering scenarios. In this paper, we propose a spatio-temporal dual denoising framework for money laundering detection on evolving graphs, named EvoDen. For spatial denoising, instead of relying on passive neighborhood aggregation, we design a risk-potential guided walk mechanism to extract denoised and suspicion-biased transaction sequences, enabling the model to isolate anomalous fund flows from noisy local neighborhoods. For temporal denoising, we design a continual Transformer-based sequence learning method driven by a reconstruction objective, which learns denoised sequence representations while enabling latent-space replay of historical knowledge to mitigate representation drift. We conduct experiments on three transaction datasets, and the results show that EvoDen outperforms state-of-the-art baselines, improving the F1-score by 2.7%~18.4% in money laundering detection. The source code is publicly available at https://github.com/AI4FinSec/EvoDen.

1 Introduction

Money laundering poses significant threats to financial stability and even fuels other serious crimes such as smuggling, drug trafficking, and corruption. Developing effective money laundering detection methods is vital to safeguarding national security. As criminals continually refine their tactics to evade regulatory scrutiny, money laundering activities have become increasingly concealed, posing significant challenges to *Anti-Money Laundering* (AML) efforts.

Through the analysis of real-world data, we find that money laundering activities are primarily concealed by criminals who exploit the structural complexity and temporal evolution of transaction networks. Specifically, criminals typically split illegal funds into multiple small transactions that resemble normal activities, subsequently concealing these illicit transactions within a large volume of legitimate transactions. Fig. 1 visualizes a Bitcoin transaction graph crawled from

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the blockchain[2], where each node represents a transaction and each edge indicates bitcoin flow between transactions. Red nodes denote laundering transactions and blue nodes denote normal transactions. As shown in the magnified neighborhood, most illicit transactions are concealed within dense clusters of legitimate activities, causing weak laundering signals to be spatially diluted. Meanwhile, laundering activities are commonly dispersed over extended time spans, causing early suspicious cues to be further obscured by subsequent transactions as the graph evolves.

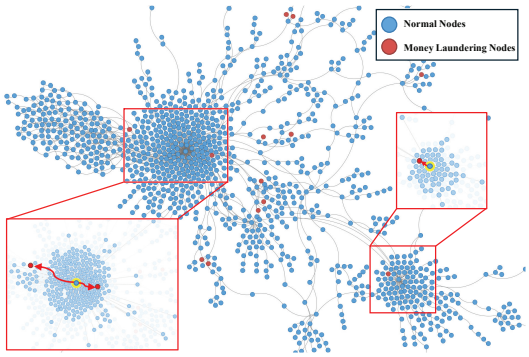


Figure 1: The visualization of a real-world transaction graph.

Despite progress in AML research, existing methods remain limited in addressing these concealment patterns. Rule-based and classical machine learning methods typically rely on predefined thresholds or hand-crafted statistical features, making them fragile when laundering transactions are intentionally shaped to resemble normal behavior Senator et al. [1995], Rajput et al. [2014], Khanuja and Adane [2014], Tang and Jian [2005], Wang and Yang [2007], Gao and Mao [2007], Gao [2009], Yang et al. [2014], Chen et al. [2014], Jensen and Iosifidis [2023]. Graph learning methods further exploit structural dependencies in transaction networks and have become effective tools for AML Michalak and Korczak [2011], Soltani et al. [2016], Li et al. [2020], Chai et al. [2023], Lin et al. [2024a], Weber et al. [2019], Cheng et al. [2023], Song et al. [2025], Xiang et al. [2025], Wang and Zhu [2025]. However, most of them still rely on neighborhood aggregation, which lacks sufficient selectivity when merging information from neighbors. As a result, illicit signals hidden within dense legitimate transaction neighborhoods can be diluted, further aggravating the over-smoothing effect.

To counter sophisticated concealment tactics deployed across both structural and temporal dimensions, we propose EvoDen, a spatio-temporal dual denoising framework for money laundering detection on evolving graphs. At the spatial structure level, we observe that money laundering is essentially a disguised process of fund flow, in which transactions between entities form an ordered sequence with contextual associations along the flow path. Hence, instead of passively inheriting noisy neighborhood information, we strategically distill critical transaction flows from complex transaction networks to model laundering behaviors. Specifically, we propose a risk-potential guided walk mechanism. This mechanism integrates individual transactional characteristics and bidirectional topological influence into a unified potential field, which explicitly guides the walk toward nodes with higher laundering-related potential. This mechanism imposes constraints on the walking process to extract critical sequences that capture abnormal fund flows. **As a result, EvoDen is able to isolate weak but essential money laundering signals along the spatial structural dimension.**

At the temporal evolution level, early-stage laundering patterns can be easily obscured by newly added transaction data. Therefore, the model must continually incorporate new knowledge while preserving historical information essential for detecting early-stage anomalies. To meet this requirement, we propose a continual Transformer-based sequence learning method driven by a reconstruction objective. The reconstruction objective first allows the model to learn the rich contextual associations embedded in denoised transaction sequences without relying on labeled data, which is a critical property in AML scenarios where illicit cases are scarce. It also naturally supports latent-space generative replay. Building on this mechanism, we replay representative historical sequences during training, which preserves the distribution of historical behaviors and prevents the latent representations from gradually drifting toward benign-dominated patterns as the transaction network evolves. **In this way, EvoDen mitigates temporal noise and enhances the ability to detect covert money laundering activities.**

The main contributions of the paper are summarized as follows:

- We reformulate money laundering detection as a spatio-temporal denoising task over evolving transaction graphs, addressing the spatial and temporal obfuscation strategies commonly employed by criminals.

[2]https://www.walletexplorer.com

- We propose a risk-potential guided sequence extraction method. Using our constructed AML-specific potential field, EvoDen extracts suspicion-biased denoised transaction sequences, reducing interference from criminals' deliberate spatial obfuscation.
- We design a continual Transformer-based sequence learning model to dynamically capture hidden contextual associations within the denoised sequences. This model preserves knowledge of previously observed laundering behaviors, enabling it to mitigate the temporal noise caused by the scattering of illicit activities across time.
- We conduct extensive experimental evaluations on three transaction datasets to demonstrate the effectiveness of EvoDen. The results show that EvoDen achieves higher F1 score in money laundering detection compared to state-of-the-art baselines.

2 Related Work

Traditional AML methods are based on rules. For example, FATF, one of the most influential international AML organizations in the world, formulated 40 policy standards for global AML Senator et al. [1995]. Rajput *et al.* proposed an ontology-based expert-system for detecting suspicious transactions, which deduced new knowledge from the predefined rules related to suspicious transactions Rajput et al. [2014]. However, rule-based AML methods can be easily evaded by criminals. Hence, machine learning algorithms Qiu and Qiu [2020], Zhang et al. [2023], Qiu et al. [2006] are applied for detecting money laundering activities intelligently. For example, Tang *et al.* argue that comparison with normal transaction behavior is needed to detect suspicious transaction behavior. Therefore, the traditional filtering system based on pre-defined rules was replaced by support vector machines based on statistical theory to detect money laundering transactions Tang and Jian [2005]. Wang *et al.* applied the decision tree algorithm for AML Wang and Yang [2007]. Gao *et al.* conducted pioneering research on money laundering, and used clustering and outlier mining to identify suspicious transactions in money laundering accounts Gao and Mao [2007], Gao [2009].

Machine learning-based AML approaches mentioned above can mine the features of users or transactions, but these methods ignore the structural features of interactions between users. Therefore, graph mining approaches are introduced into the AML field by modeling financial transaction networks as graphs. Ketenci *et al.* proposed a fuzzy matching method of graph structures by using fuzzy numbers which represent parameters of transactions Michalak and Korczak [2011]. Soltani *et al.* detected money laundering activities by utilizing structural similarity to cluster transactions Soltani et al. [2016]. Shi *et al.* proposed to model transaction networks as multipartite graphs, and detected the complete flow of money from source to destination using a scalable algorithm, FlowScope Li et al. [2020]. Chai *et al.* proposed an outstanding AML framework to discriminate money laundering transactions from benign transactions and capture multi-hop underlying patterns of money laundering sub-network Chai et al. [2023]. Lin *et al.* proposed an innovative DenseFlow framework that effectively identifies and traces money laundering activities by finding dense subgraphs and applying the maximum flow idea Lin et al. [2024a]. Moreover, graph neural networks are considered as one of the most effective methods to detect money laundering networks. For example, Weber *et al.* contributed a Bitcoin transaction dataset, Elliptic. They modeled Bitcoin transaction data as a graph and utilized graph convolutional networks to detect money laundering transactions Weber et al. [2019]. Cheng *et al.* proposed a group-aware deep graph learning method for detecting money laundering groups by integrating topological structure and attribute information to generate node representations Cheng et al. [2023]. Song *et al.* proposed a social attention graph neural network on account graphs to detect money laundering accounts Song et al. [2025]. Xiang *et al.* proposed an attribute-driven temporal graph neural network for detecting illicit activities Xiang et al. [2025]. Wang *et al.* designed a graph learning-based model to learn the features of transaction attributes, events, and entities for detecting illicit activities in online payment systems Wang and Zhu [2025]. To better handle large-scale and complex financial transaction networks, Lin *et al.* proposed FraudGT, a graph transformer framework that incorporates edge-based message passing and edge-attribute attention bias to efficiently detect financial fraud on large-scale transaction graphs Lin et al. [2024b]. Besides, we provide additional discussion of continual graph learning methods in Appendix B.

However, existing AML and graph learning methods remain limited in selectively isolating suspicious fund-flow signals and preserving historical cues under graph evolution, which motivates us to develop a spatio-temporal dual denoising framework.

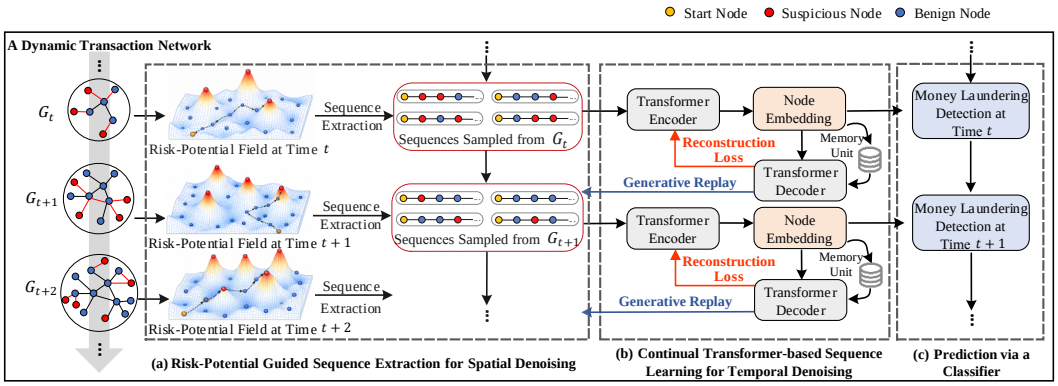


Figure 2: The framework of EvoDen.

3 Problem Descriptions and EvoDen Framework

Dynamic Transaction Graphs. A transaction is a fund-transfer event that moves monetary value from one financial entity to another, and a transaction network is therefore viewed as a directed graph describing how funds propagate through the system. Depending on the data source, the graph elements may have different semantics. In account-level financial networks, nodes usually represent accounts and directed edges denote transaction events. In Bitcoin transaction networks, nodes can denote transactions and directed edges denote the flow of bitcoins between transactions. Although these representations differ in form, both characterize the directed movement of funds. Formally, we denote the transaction graph at time t as $G_t = (V_t, E_t)$, where V_t and E_t are the graph vertices and directed edges under the corresponding data representation. For modeling convenience, let $\mathcal{X}_t$ denote the set of transaction events observed up to time t. A dynamic transaction graph evolves incrementally over time and can be represented as $\mathcal{G} = (G_1, G_2, ..., G_t)$, where each state G_t is constructed based on the previous state G_{t-1} with incremental updates $\triangle G_t$. That is $G_t = G_{t-1} + \triangle G_t$.

Incremental Transactions. An incremental transaction refers to a newly observed fund-transfer event at the current timestamp.

Transaction Sequences. In EvoDen, we extract critical transaction sequences centered on each target transaction event using the risk-potential guided walk mechanism. These sequences are sampled from the neighborhood of the target transaction and designed to capture suspicious fund flows while filtering out noise. For a target transaction event $x_i \in \mathcal{X}_t$ at time t, the extracted denoised sequence is denoted as $L_i = \{x_i, x_{i1}, x_{i2}, \ldots, x_{i(l-1)}\}$, where $x_{i1}, x_{i2}, \ldots, x_{i(l-1)}$ are neighbor transaction events arranged in walking order. The feature of L_i is denoted by $F_i = [f_i^T, f_{i1}^T, f_{i2}^T, ...]^T$, where f_i represents the raw feature of x_i, and $f_{i1}, f_{i2}, ...$ represent the raw features of these neighboring transactions. In Bitcoin transaction graphs, these transaction events correspond to transaction nodes. In account-level transaction graphs, they correspond to transaction edges traversed along the walking path. Thus, regardless of the raw graph representation, the input to the sequence learner is always an ordered transaction sequence.

Problem Descriptions. The AML problem to be solved in this paper is described as follows: given a dynamic transaction network $\mathcal{G}$, our goal is to infer the likelihood of incremental transactions being involved in money laundering at each timestep.

The Framework of EvoDen. To describe the framework of EvoDen clearly, we depict the detection procedures in Fig. 2. To adapt to the dynamic changes of transaction networks, we do not repeat the model training on the whole network at each timestamp. Instead, we update the model using only incremental data and a small quantity of replay data. At each timestep, the risk-potential guided walking method is first employed to extract critical transaction sequences that capture anomalous transaction flows for each incremental node, thereby achieving spatial denoising. Then, the continual Transformer-based sequence learning method is utilized to learn the hidden contextual information within these sequences and to generate sequence embeddings, thereby achieving temporal denoising. In the continual Transformer model, considering the high cost of label acquisition, we design a reconstruction loss to enable learning transaction embeddings without reliance on labels, as indicated by the red arrow ('$\rightarrow$'). To avoid catastrophic forgetting when handling dynamically evolving

transaction data, we propose a clustering-based latent-space generative replay method to retain learned knowledge and enhance the model's robustness against distribution shifts, as indicated by the blue arrow ('$\rightarrow$'). Then, the transaction embeddings learned by the continual Transformer are fed into a classifier to detect money laundering activities.

4 The Proposed EvoDen Method

4.1 Risk-potential Guided Sequence Extraction for Spatial Denoising

We first extract critical neighborhood information for each incremental transaction to be detected from its noisy surrounding context. Although *Random Walks with Restart* (RWRs) can effectively characterize node neighborhoods Wang et al. [2022b], uniform neighbor sampling is less suitable for AML scenarios, where suspicious fund-flow cues are often diluted by dense legitimate transactions. Therefore, we design a risk-potential guided walk that biases the transition toward more suspicious fund-flow paths. Specifically, we define a risk-potential field over the transaction graph, where each node is assigned a scalar potential score $\varphi(\cdot)$ to indicate its laundering-related risk in the local fund-flow structure. This field provides a unified view for guiding the walk toward more suspicious regions in the transaction neighborhood. For Bitcoin transaction graphs, the potential score is defined on transaction nodes. For account-level transaction graphs, the potential score is defined on account nodes and is used to guide the selection of transaction edges along the walking path.

The potential score of node i consists of an individual potential $\varphi_{\triangle}(i)$, determined by its own transaction behavior, and a topological potential $\varphi_{\circ}(i)$, influenced by interactions with other nodes. It is denoted by $\varphi(i) = \varphi_{\triangle}(i) + \varphi_{\circ}(i)$. For individual potential, we use transaction statistics to describe the behavioral intensity of each node. The individual potential of node i is defined as $\varphi_{\triangle}(i) = \left(\prod_{q=1}^{3} \delta_q(i)\right)^{\frac{1}{3}}$. Here, $\delta_1(i), \delta_2(i), \delta_3(i)$ denote the normalized transaction frequency, average transaction amount and expenditure-to-income ratio of node i, respectively. These behavioral indicators are potentially transferable across different financial systems because they describe generic transaction behaviors—such as activity intensity, monetary scale, and fund-flow imbalance—rather than platform-specific patterns, and can be derived from commonly available transaction attributes. For financial networks with different transaction semantics or available attributes, these indicators can be recalibrated, replaced, or augmented without changing the overall EvoDen framework.

For topological potential, we define the interaction between nodes using a bidirectional topological potential. For simplicity, we only consider nodes within three hops of a given node. The bidirectional topological potential of node i is calculated as $\varphi_{\circ}(i) = TP_{in}(i) + TP_{out}(i)$. Here, $TP_{in}(i)$ represents the topological potential based on fund convergence, indicating the extent to which node i is influenced by other nodes. $TP_{out}(i)$ is defined as the topological potential based on fund divergence, representing the ability of node i to influence other nodes. The definitions of both are given as follows:

$$TP_{in}(i) = \sum_{j \in V_{in}^i} \frac{a^{out}(j)}{deg^{out}(j)} e^{-\left(\frac{d_{ji}}{\sigma}\right)^2}, \quad TP_{out}(i) = \sum_{j \in V_{out}^i} \frac{a^{out}(i)}{deg^{out}(i)} e^{-\left(\frac{d_{ij}}{\sigma}\right)^2}, \tag{1}$$

where $a^{out}(\cdot)$ represents the total expenditure of the node. $deg^{out}(\cdot)$ denotes the out-degree of the node. d_{ji} represents the distance from node j to node i. V_{in}^i denotes the set of neighboring nodes that have a reachable path to node i within a three-hop range. V_{out}^i represents the set of nodes that are reachable from node i within a three-hop range. The influence factor σ represents the range of influence of the node. Its value can be obtained by optimizing the topological potential entropy as follows:

$$\min \ Entropy(\sigma) = -\sum_{i=1}^{|V^i|} \frac{\varphi_{\circ}(i)}{\mathcal{Z}} \log\left(\frac{\varphi_{\circ}(i)}{\mathcal{Z}}\right), \tag{2}$$

where $\mathcal{Z} = \sum_{i=1}^{|V^i|} \varphi_{\circ}(i)$. $|V^i|$ represents the set of all nodes within three hops of node i. Minimizing the topological-potential entropy highlights structurally influential nodes during the guided walk. We employ a stochastic gradient descent algorithm for computation. To reduce the time cost of iterative calculations, we constrain the value of σ to the range $(0, 3]$.

Given a target transaction event x_i, EvoDen performs a risk-potential guided walk to extract a denoised transaction sequence L_i of length l. The walk restarts from the start node with probability α

and otherwise moves to a neighboring node sampled with the potential score as the transition weight

$$p(j \mid i) = (1-\alpha)\frac{\varphi(j)}{\sum_{k\in\mathcal{N}(i)}\varphi(k)}, \quad j \in \mathcal{N}(i), \tag{3}$$

where i is the current node, j is a candidate neighboring node, and $\mathcal{N}(i)$ is the neighborhood of i. In Bitcoin transaction graphs, the walk starts from the target transaction node x_i, and visited transaction nodes are recorded in L_i. In account-level transaction graphs, the walk starts from the receiver account of x_i to follow subsequent fund flows, and the traversed transaction edges are recorded in L_i. The resulting sequence is denoted as $L_i = \{x_i, x_{i1}, x_{i2}, \ldots, x_{i(l-1)}\}$, with feature matrix $F_i = [\mathbf{f}_i^T, \mathbf{f}_{i1}^T, \mathbf{f}_{i2}^T, \ldots, \mathbf{f}_{i(l-1)}^T]^T$. The detailed walking procedure is provided in Appendix C.1.

4.2 Continual Transformer-based Sequence Learning for Temporal Denoising

After obtaining denoised transaction sequences, EvoDen learns their contextual representations with a continual Transformer-based sequence learner. Given the feature sequence F_i of a target transaction event, we add positional encodings and feed it into a Transformer encoder

$$\widetilde{F}_i = \mathrm{Enc}(F_i), \tag{4}$$

where $\widetilde{F}_i$ denotes the learned sequence representation. To train the sequence learner without relying on dense AML labels, EvoDen uses a reconstruction objective, where a Transformer decoder reconstructs the input transaction sequence from its latent representation

$$\widehat{F}_i = \mathrm{Dec}(\widetilde{F}_i), \tag{5}$$

$$\mathcal{L}_{rec} = \frac{1}{M}\sum_{i=1}^{M}\left\|\widehat{F}_i - F_i\right\|^2, \tag{6}$$

where M denotes the number of transaction sequences used in the update step.



Figure 3: The architecture of the continual Transformer-based sequence learning model.

After sequence learning, the encoder output $\widetilde{F}_i$ is aggregated into a transaction-level representation for downstream money laundering detection. Specifically, the representation of target transaction event x_i is calculated as

$$\boldsymbol{F}_i = \widetilde{f}_i + \sum_{j=1}^{l-1}\frac{\widetilde{f}_{ij}}{d_{ji}}, \tag{7}$$

where $\widetilde{f}_i$ denotes the learned representation of the target transaction event, $\widetilde{f}_{ij}$ denotes the learned representation of the j-th transaction event in the sequence, and d_{ji} denotes its hop distance to x_i. To mitigate temporal noise as the transaction graph evolves, EvoDen performs clustering-based latent-space replay. After each update, we cluster the learned transaction representations and select representative embeddings from each cluster into a memory unit

$$\{C_1, \ldots, C_\kappa\} = \mathrm{KMeans}(\{\boldsymbol{F}_i\}_{i=1}^{M}). \tag{8}$$

The cluster centroids are used only for selecting representative sample embeddings, which are stored in memory and refreshed through reclustering after each update. To avoid manually tuning the cluster number, we determine κ automatically by maximizing the silhouette coefficient over a small candidate range. This clustering step encourages the memory to cover diverse historical transaction patterns rather than retaining only recent or frequent ones. During subsequent updates, the stored embeddings are decoded into replayed sequence features

$$F_i^{\mathrm{replay}} = \mathrm{Dec}(\widetilde{F}_i^{\mathrm{mem}}), \quad \widetilde{F}_i^{\mathrm{mem}} \in \mathcal{M}, \tag{9}$$

where $\widetilde{F}_i^{\mathrm{mem}}$ denotes a stored latent sequence representation in the memory unit $\mathcal{M}$. The replayed features are combined with newly extracted sequences for continual training, helping EvoDen preserve historical laundering cues that may otherwise be diluted by incoming benign transactions. Detailed continual Transformer architecture is provided in Appendix C.2.

4.3 Money Laundering Detection

The aggregated representation $\boldsymbol{F}_i$ is then fed into a two-layer MLP followed by a sigmoid function to predict the probability that transaction x_i is involved in money laundering. We train the classifier with the binary cross-entropy loss

$$\mathcal{L} = -\frac{1}{|Z|}\sum_{z=1}^{|Z|}\left[y(z)\log\hat{y}(z) + (1-y(z))\log(1-\hat{y}(z))\right], \tag{10}$$

where Z denotes the set of samples in each batch, $|Z|$ is the batch size, $y(z)$ is the ground-truth label, and $\hat{y}(z)$ is the predicted laundering probability. During training, we use the Adam Optimizer to learn the model parameters. The learning rate is set to 0.001.

5 Experiments and Performance Analyses

We organize the experiments around the following research questions. **RQ1: Overall detection performance.** Can EvoDen achieve strong money laundering detection performance under both static and dynamic transaction graph settings compared with classical machine learning, graph learning, domain-specific AML methods, and continual learning baselines? **RQ2: Scalability and economic utility.** Can EvoDen generalize to large-scale and highly imbalanced AML benchmarks, and can it recover a larger fraction of illicit monetary volume? **RQ3: Component contribution.** How much do the risk-potential guided sequence extraction and the latent-space replay mechanism contribute to the final performance? **RQ4: Complementarity between spatial and temporal denoising.** To what extent can reconstruction-based sequence learning compensate for degraded spatial sampling, and how do spatial and temporal denoising complement each other? In addition, we provide further analyses, including parameter sensitivity analyses, efficiency evaluation under different data increment scales, and a case study in Appendices D.6, D.7 and D.8, respectively.

5.1 Datasets and Experiment Settings

Datasets. We evaluate EvoDen on three transaction datasets, including two Bitcoin transaction datasets, Elliptic Weber et al. [2019] and WalletExplorer, and the large-scale IBM-AML benchmark Altman et al. [2023]. Elliptic is a public Bitcoin transaction graph, while WalletExplorer is a Bitcoin transaction dataset we collect from the WalletExplorer blockchain explorer[3]. Both Bitcoin datasets represent transaction events as nodes, whereas IBM-AML provides account-level transaction records with multiple scales and illicit ratios. Together, these datasets cover both real-world blockchain transactions and large-scale simulated AML scenarios. Detailed dataset statistics, feature construction, and preprocessing procedures are provided in Appendix D.1.

Baseline Methods. We compare EvoDen with 15 baselines covering classical machine learning, graph learning, domain-specific AML detection, and dynamic or continual learning methods. Classical machine learning baselines include LR Hosmer and Lemeshow [2000], RF Breiman [2001], and XGB Chen and Guestrin [2016]. Graph learning and domain-specific AML baselines include LINE Tang et al. [2015], GCN Alarab et al. [2020], GraphSAGE Hamilton et al. [2017], Inspection-L Lo et al. [2023], and FraudGT Lin et al. [2024b]. Dynamic and continual learning baselines include DNE Du et al. [2018], SingleGNN, OnlineGNN, GNN-EWC Kirkpatrick et al. [2017], GNN-ER Rolnick et al. [2019], GRANDE Wu et al. [2022], and HAMLET Tatulli et al. [2023]. Detailed descriptions of the baselines are given in Appendix D.2.

Implementation. For implementation, our hardware system includes a server containing one CPU with an Intel Xeon Gold 5220, two Tesla V100 GPUs, and 32GB of memory. The software environment is based on Python 3.9. All methods are evaluated under the same data splits and evaluation protocol. Each experiment is repeated 10 times, and we report the mean performance with standard deviation. We use precision, recall, F1-score, AUC (area under the ROC curve) and MCC as evaluation metrics. For IBM-AML, we additionally report the Recovered Laundering Amount Ratio (RLAR), following the asset recovery objective emphasized by FATF Financial Action Task Force [2023]. RLAR is defined as the laundering monetary volume associated with correctly detected illicit transactions divided by the total laundering volume in the test set. To assess the robustness of the

[3]https://www.walletexplorer.com

Table 1: Performance results for laundering transaction detection on Elliptic and WalletExplorer.

Method	Elliptic			WalletExplorer		
	F1 ↑	AUC ↑	MCC ↑	F1 ↑	AUC ↑	MCC ↑
LR	0.4422±0.0000	0.8983±0.0000	0.4272±0.0000	0.5689±0.0000	0.7554±0.0000	0.4852±0.0000
RF	0.8219±0.0039	0.9195±0.0041	0.8204±0.0046	0.8124±0.0079	0.8429±0.0089	0.7753±0.0111
XGB	0.8058±0.0000	0.9211±0.0000	0.7997±0.0000	0.7970±0.0000	0.8343±0.0000	0.7520±0.0000
LINE	0.6711±0.0079	0.7487±0.0019	0.6553±0.0088	0.6469±0.0131	0.8003±0.0040	0.5627±0.0208
GCN	0.7858±0.0055	0.8201±0.0014	0.7799±0.0057	0.7763±0.0064	0.8036±0.0059	0.7300±0.0124
GraphSAGE	0.8830±0.0038	0.9095±0.0071	0.8752±0.0040	0.8555±0.0089	0.8807±0.0035	0.8224±0.0176
Inspection-L	0.7816±0.0095	0.8667±0.0040	0.7734±0.0104	0.8041±0.0037	0.8203±0.0113	0.7614±0.0082
FraudGT	0.8852±0.0066	0.9289±0.0036	0.8641±0.0075	0.8617±0.0071	0.8842±0.0022	0.8300±0.0086
EvoDen (Static)	**0.9347******±0.0050**	**0.9371******±0.0023**	**0.9306******±0.0068**	**0.8887******±0.0091**	**0.9092******±0.0058**	**0.8634******±0.0125**
DNE	0.6419±0.0115	0.6762±0.0178	0.6133±0.0089	0.6351±0.0057	0.6030±0.0033	0.6310±0.0040
SingleGNN	0.6713±0.0094	0.7239±0.0065	0.6551±0.0073	0.6865±0.0064	0.7332±0.0021	0.6649±0.0097
OnlineGNN	0.7411±0.0098	0.7992±0.0045	0.7393±0.0070	0.7293±0.0082	0.7698±0.0045	0.7122±0.0050
GNN-EWC	0.8183±0.0074	0.8334±0.0077	0.8120±0.0069	0.8040±0.0088	0.8175±0.0062	0.7816±0.0143
GNN-ER	0.8079±0.0057	0.8209±0.0079	0.7822±0.0084	0.8279±0.0097	0.8326±0.0046	0.7930±0.0081
GRANDE	0.7146±0.0016	0.7896±0.0055	0.7031±0.0013	0.6939±0.0081	0.8015±0.0060	0.6911±0.0028
HAMLET	0.6131±0.0052	0.6908±0.0058	0.5799±0.0070	0.5980±0.0094	0.7018±0.0111	0.5778±0.0084
EvoDen (Dynamic)	**0.9075******±0.0055**	**0.9126******±0.0024**	**0.8902******±0.0033**	**0.8688******±0.0071**	**0.8536******±0.0052**	**0.8936******±0.0070**

improvements, we conduct a paired t-test between EvoDen and the second-best method over the 10 repeated runs. The symbols $*$ and $**$ indicate statistically significant improvements at the $p < 0.05$ and $p < 0.01$ levels, respectively.

5.2 RQ1: Overall Detection Performance

We evaluate the overall effectiveness of EvoDen under both static and dynamic transaction graph settings. In the static setting, EvoDen is trained without replay on the whole transaction graph split by time, which evaluates whether the risk-potential guided sequence extraction and reconstruction-based sequence learning can produce effective transaction representations. In the dynamic setting, models are updated with newly arriving transactions, which evaluates whether EvoDen can maintain strong detection performance as the transaction graph evolves. Detailed data splits, model configurations, and training protocols are provided in Appendix D.3. Table 1 reports the key results on Elliptic and WalletExplorer. To save space, we provide the complete results with all evaluation metrics in Appendix D.4. In the static setting, EvoDen consistently outperforms classical machine learning methods, graph learning methods, and domain-specific AML baselines across the main evaluation metrics. Compared with Inspection-L, EvoDen achieves average relative improvements of 15.06% in F1-score and 16.86% in MCC on the two datasets. Compared with the strong baseline FraudGT, EvoDen further improves F1-score and MCC by 4.36% and 5.86% on average, respectively. The paired t-test results show that these gains are statistically significant, indicating that the improvement is stable across repeated runs. To provide a more intuitive interpretation of the detection results, we provide confusion matrices on Elliptic in Appendix D.4. In the dynamic setting, EvoDen also achieves the best overall performance compared with dynamic and continual learning baselines such as DNE, OnlineGNN, GNN-EWC, GNN-ER, GRANDE, and HAMLET. This demonstrates that the proposed temporal denoising module can incorporate newly arriving transaction patterns while preserving historical laundering cues.

5.3 RQ2: Scalability and Economic Utility

We evaluate EvoDen on the IBM-AML benchmark to examine its scalability and practical utility under large-scale and highly imbalanced AML scenarios. IBM-AML contains six configurations with different graph scales and illicit ratios, ranging from Small to Large and from higher-illicit to lower-illicit settings. We follow the experimental setup of FraudGT Lin et al. [2024b] for consistency. Detailed settings and additional analyses are provided in Appendix D.5. Table 2 reports F1-score and RLAR on IBM-AML. Traditional classifiers perform poorly under extreme class imbalance, while graph-based baselines improve the results by exploiting transaction structures. EvoDen achieves consistently strong performance across different scales and illicit ratios, showing the benefit of spatio-temporal dual denoising on large transaction graphs. In the static setting, EvoDen obtains near-best F1 and RLAR on most configurations. Notably, on the challenging Large-LI split, EvoDen

Table 2: Detection performance and economic benefit on IBM-AML datasets.

Methods	AML Small-HI		AML Small-LI		AML Medium-HI		AML Medium-LI		AML Large-HI		AML Large-LI	
	F1 ↑	RLAR ↑	F1 ↑	RLAR ↑	F1 ↑	RLAR ↑	F1 ↑	RLAR ↑	F1 ↑	RLAR ↑	F1 ↑	RLAR ↑
LR	0.0054	0.0087	0.0011	0.0045	0.0008	0.0000	0.0029	0.0005	0.0056	0.0042	0.0041	0.0025
RF	0.1927	0.2766	0.0238	0.0824	0.1929	0.0186	0.0550	0.0084	0.2102	0.1200	0.0647	0.0431
XGB	0.1831	0.2502	0.0039	0.0112	0.2107	0.0209	0.0052	0.0009	0.1381	0.0980	0.0008	0.0004
LINE	0.3018	0.4940	0.1072	0.3588	0.3584	0.3015	0.0674	0.0096	0.2310	0.1559	0.0295	0.0122
GCN	0.3920	0.5324	0.1550	0.5897	0.4428	0.0284	0.1033	0.0130	0.3857	0.2511	0.0929	0.0576
GraphSAGE	0.2573	0.3102	0.0831	0.1660	0.3610	0.0209	0.0528	0.0085	0.2430	0.2099	0.0512	0.0447
Inspection-L	0.3217	0.6215	0.1136	0.4104	0.3904	0.2411	0.0967	0.0388	0.2701	0.3866	0.0638	0.0400
FraudGT	0.7549	0.7511	0.4592	0.7970	0.7613	0.3158	0.4255	0.1672	0.7443	0.4903	**0.3822**	0.2104
EvoDen (Static)	**0.7984****	**0.7990****	**0.4918****	**0.8180****	**0.7850****	**0.3569****	**0.4521****	**0.1935****	**0.7597****	**0.5500****	0.3601	**0.2515****
DNE	0.2551	0.3730	0.0623	0.2823	0.2418	0.1955	0.0274	0.0117	0.2036	0.1731	0.0125	0.0087
SingleGNN	0.2319	0.2964	0.0846	0.2119	0.2525	0.2143	0.0402	0.0535	0.1920	0.1600	0.0207	0.0163
OnlineGNN	0.2783	0.3521	0.1077	0.3002	0.3092	0.3210	0.0637	0.0094	0.2471	0.1962	0.0419	0.0194
GNN-EWC	0.2811	0.3368	0.1340	0.4846	0.3674	0.2958	0.0891	0.0100	0.2904	0.2022	0.0563	0.0370
GNN-ER	0.3299	0.5011	0.1328	0.4385	0.3861	0.2889	0.0920	0.0631	0.3385	0.2114	0.0776	0.0518
GRANDE	0.4820	0.6480	0.2449	0.5551	0.4988	0.3926	0.1817	0.1157	0.5230	0.4009	0.2381	0.1702
HAMLET	0.2541	0.4102	0.1565	0.4240	0.3712	0.3504	0.1001	0.0868	0.3413	0.2950	0.1990	0.1983
EvoDen (Dynamic)	**0.6518****	**0.7063****	**0.4003****	**0.6916****	**0.6241****	**0.4221****	**0.3655****	**0.1570****	**0.6512****	**0.4498****	**0.3101****	**0.2234****

achieves a higher RLAR even when its F1-score is slightly lower than FraudGT, suggesting that it identifies laundering activities associated with larger illicit monetary volume. In the dynamic setting, EvoDen also maintains competitive performance as new transactions arrive, demonstrating its ability to preserve useful laundering cues during graph evolution.

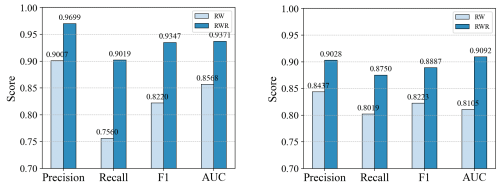
To further examine the scalability of the three-hop potential computation to large graphs, we conduct an additional sensitivity analysis on AML Large-LI, which contains 2,070,980 nodes and 176,066,557 directed edges. We vary the topological potential range from one to five hops while keeping the walk length, model architecture, and all other settings unchanged, and report both F1-score and average topological potential computation time per node. The results are shown in Table 3. The results show that expanding the range from one to three hops improves the F1-score from 0.3400 to 0.3601 by incorporating informative multi-hop fund-flow patterns. Although four hops achieves a slightly higher F1-score of 0.3628, the improvement over three hops is only 0.0027, while the average computation time increases by approximately 2.99 times. Extending the range to five hops provides no further improvement over four hops and requires approximately 8.61 times the computation time of three hops. These results support three hops as a practical trade-off between effectiveness and efficiency.

Table 3: Scalability of the topological potential range on AML Large-LI.

Hop Range	F1	Average Computation Time per Node (ms)
1	0.3400	0.030
2	0.3467	0.094
3	0.3601	0.283
4	0.3628	0.846
5	0.3609	2.438

5.4 RQ3: Component Contribution

In EvoDen, three core components are contained: 1) the risk-potential guided sequence extraction method, 2) the continual Transformer-based sequence learning method, and 3) the clustering-based latent-space generative replay method. The previous sections highlight the benefits of the continual Transformer-based sequence learning method. In this section, we discuss how the other components contribute to enhancing the performance of EvoDen. In the risk-potential guided sequence extraction method, we discuss the performance impact of the walk algorithm. We replace the walk method in EvoDen with a simple random walk method (RW). The comparison of performance results is shown in Fig. 4. Our walk method outperforms RW across all metrics on both datasets, with average 10.89%↑ and 10.77%↑ improvement respectively in F1-score and AUC. This demonstrates the effectiveness of our sequence extraction method. Since our walk algorithm



(a) Elliptic (b) WalletExplorer

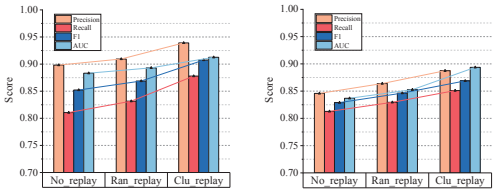
Figure 4: Performance results under different random walking methods.

Table 4: Complementarity between spatial and temporal denoising on Elliptic.

Walk Method	Sequence Representation	Precision	Recall	F1	AUC
Random walk	Mean pooling	0.8365	0.7473	0.7894	0.8235
Random walk	Reconstruction-based Transformer	0.9007	0.7560	0.8220	0.8568
Risk-potential guided walk	Mean pooling	0.8502	0.8296	0.8398	0.8709
Risk-potential guided walk	Reconstruction-based Transformer	0.9699	0.9019	0.9347	0.9371

fully considers characteristics related to money laundering, the neighboring nodes generated by the walk method provide references for making decisions in our AML model. To some extent, the walk algorithm we proposed also provides an explanation for the classification results of EvoDen.

Then, we evaluate the impact of the clustering-based latent-space generative replay method. We compare three variants of EvoDen: without replay (No_replay), with random replay (Ran_replay), and with clustering-based replay (Clu_replay). As shown in Fig. 5, Clu_replay consistently outperforms the other two variants. Although Ran_replay benefits from incorporating historical samples, its random selection strategy may fail to preserve diverse historical patterns. In contrast, Clu_replay performs K-means clustering in the embedding space learned by the continual Transformer and selects balanced samples from different clusters. This enables EvoDen to retain more representative historical behaviors and remain sensitive to earlier or rarer laundering patterns as the transaction graph evolves.



(a) Elliptic (b) WalletExplorer

Figure 5: Performance results under different replay methods.

5.5 RQ4: Complementarity Between Spatial and Temporal Denoising.

To investigate whether temporal denoising can compensate for degraded spatial sampling and to further clarify the complementarity between the two denoising stages, we conduct a controlled ablation study by combining different walk strategies and sequence representation methods. We compare random walk and risk-potential guided walk, each combined with mean pooling or the reconstruction-based Transformer, while keeping all other settings unchanged. The results are shown in Table 4. With random walk, replacing mean pooling with the reconstruction-based Transformer improves F1 from 0.7894 to 0.8220, indicating that temporal sequence learning can partially compensate for noisy spatial sampling. However, this variant remains inferior to the risk-potential guided walk with mean pooling, which achieves an F1-score of 0.8398, suggesting that information omitted during sequence extraction cannot be fully recovered at the representation-learning stage. Moreover, applying the reconstruction-based Transformer to risk-potential guided sequences further raises F1 to 0.9347. Overall, spatial denoising improves the retained transaction context, while temporal denoising further models its dependencies; their combination yields the strongest performance.

6 Conclusion

In this paper, we proposed a novel AML method called EvoDen to detect concealed money laundering activities via spatio-temporal dual denoising and enhance the financial system's ability to control laundering risks. Through the analysis of real-world data, we found that criminals typically conceal money laundering transactions by exploiting both the structural complexity and temporal evolution of the transaction network. To address this, we designed a risk-potential guided walk method to extract denoised sequences that capture abnormal fund flows. This enables EvoDen to denoise the obfuscation tactics employed by criminals along the spatial structural dimension. Based on these sequences, we further proposed a continual Transformer-based sequence learning model to explore hidden contextual associations in dynamic transaction networks, thereby denoising along the temporal evolution dimension. Extensive experiments on real-world and synthetic transaction datasets demonstrated the effectiveness of EvoDen in detecting concealed laundering activities.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Discipline Breakthrough Pioneer Program of the Ministry of Education of China (Grant No. JYB2025XDXM122), in part by the National Key Research and Development Program of China (Grant No. 2025YFB3003502), in part by the National Natural Science Foundation of China under Grant 62373094, in part by the Interdisciplinary Frontier Innovation Team Development Special Fund of Donghua University, in part by Donghua University 2024 Cultivation Project of Discipline Innovation under Grant xkcx-202406, and in part by the Fundamental Research Funds for the Central Universities under Grant CUSF-DH-T-2025. The authors declare that they have no conflicts of interest to disclose.

References

Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proc. ICMLT*, page 23–27, New York, NY, USA, 2020.

Erik Altman, Jovan Blanuša, Luc von Niederhäusern, Béni Egressy, Andreea Anghel, and Kubilay Atasu. Realistic synthetic financial transactions for anti-money laundering models. In *Proc. NIPS*, 2023.

Leo Breiman. Random Forests. *Machine learning*, 45:5–32, 2001.

Ziwei Chai, Yang Yang, Jiawang Dan, Sheng Tian, Changhua Meng, Weiqiang Wang, and Yifei Sun. Towards Learning to Discover Money Laundering Sub-network in Massive Transaction Network. In *Proc. AAAI*, pages 5363–5370, New York, USA, 2023.

Tianqi Chen and Carlos Guestrin. Xgboost: A Scalable Tree Boosting System. In *Proc. ACM SIGKDD*, pages 785–794, 2016.

Z. Chen, Dvk Le, A. Nazir, E. N. Teoh, and E. K. Karupiah. Exploration of the Effectiveness of Expectation Maximization Algorithm for Suspicious Transaction Detection in Anti-Money Laundering. In *Proc. IEEE ICOS*, pages 145–149, Subang, Selangor, Malaysia, Oct. 2014.

Dawei Cheng, Yujia Ye, Sheng Xiang, Zhenwei Ma, Ying Zhang, and Changjun Jiang. Anti-Money Laundering by Group-Aware Deep Graph Learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12444–12457, 2023.

Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding. In *Proc. IJCAI*, volume 2018, pages 2086–2092, 2018.

Financial Action Task Force. Amendments to the fatf standards to strengthen global asset recovery, 2023. URL `https://www.fatf-gafi.org/en/publications/Fatfrecommendations/amendment-FATF-standards-global-asset-recovery.html`.

Z. Gao. Application of Cluster-Based Local Outlier Factor Algorithm in Anti-Money Laundering. In *Proc. IEEE MASS*, pages 1–4, Beijing, China, Sep. 2009.

Z. Gao and Y. Mao. A Framework for Data Mining-based Anti-Money Laundering Research. *Journal of Money Laundering Control*, 10(2):170–179, 2007.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *Advances in neural information processing systems*, 30, 2017.

David W. Hosmer and Stanley Lemeshow. Applied Logistic Regression. pages 91–142, 2000.

Hu, Yining and Seneviratne, Suranga and Thilakarathna, Kanchana and Fukuda, Kensuke and Seneviratne, Aruna. Characterizing and Detecting Money Laundering Activities on the Bitcoin Network. *arXiv*, 2019.

Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. Qualifying and Raising Anti-Money Laundering Alarms with Deep Learning. *Expert Systems with Applications*, 214:119037, 2023.

Harmeet Kaur Khanuja and Dattatraya S. Adane. Forensic Analysis for Monitoring Database Transactions. In *Proc. SSCC*, pages 201–210, Taibei, China, Nov. 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Xiaoyu Kou, Yankai Lin, Shaobo Liu, Peng Li, Jie Zhou, and Yan Zhang. Disentangle-based Continual Graph Representation Learning. *ArXiv*, 2020.

X. Li, S. Liu, Z. Li, X. Han, and C. Shi. FlowScope: Spotting Money Laundering Based on Graphs. In *Proc. AAAI*, pages 4731–4738, New York, USA, Feb. 2020.

Dan Lin, Jiajing Wu, Yunmei Yu, Qishuang Fu, Zibin Zheng, and Changlin Yang. DenseFlow: Spotting Cryptocurrency Money Laundering in Ethereum Transaction Graphs. In *Proc. WWW*, pages 4429–4438, 2024a.

Junhong Lin, Xiaojie Guo, Yada Zhu, Samuel Mitchell, Erik Altman, and Julian Shun. FraudGT: A Simple, Effective, and Efficient Graph Transformer for Financial Fraud Detection. In *Proc. ICAIF*, page 292–300, New York, NY, USA, 2024b.

Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming Catastrophic Forgetting in Graph Neural Networks. In *Proc. AAAI*, volume 35, pages 8653–8661, 2021.

Wai Weng Lo, Gayan K Kulatilleke, Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Inspection-L: Self-supervised GNN Node Embeddings for Money Laundering Detection in Bitcoin. *Applied Intelligence*, pages 1–12, 2023.

K. Michalak and J. J. Korczak. Graph Mining Approach to Suspicious Transaction Detection. In *Proc. FedCSIS*, pages 69–75, Szczecin, Poland, Jul. 2011.

M. Qiu and H. Qiu. Review on Image Processing Based Adversarial Example Defenses in Computer Vision. In *Proc. BigDataSecurity*, pages 94–99, Baltimore, MD, USA, May. 2020.

M. Qiu, K. Zhang, and M. Huang. Usability in Mobile Interface Browsing. *Web Intelligence and Agent Systems*, 4(1):43–59, 2006.

Q. Rajput, N. S. Khan, A. Larik, and S. Haider. Ontology Based Expert-System for Suspicious Transactions Detection. *Computer and Information Science*, 7(1):103–114, 2014.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Ted E. Senator, Henry G. Goldberg, Jerry Wooton, Matthew A. Cottini, and Raphael W. H. Wong. Financial Crimes Enforcement Network AI System (FAIS) Identifying Potential Money Laundering from Reports of Large Cash Transactions. *AI Magazine*, 16(4):21–39, 1995.

R. Soltani, U. T. Nguyen, Y. Yang, M. Faghani, and A. An. A New Algorithm for Money Laundering Detection based on Structural Similarity. In *Proc. IEEE UEMCON*, pages 1–7, New York,USA, Oct. 2016.

Jie Song, Sijia Zhang, Pengyi Zhang, Junghoon Park, Yu Gu, and Ge Yu. Illicit Social Accounts? Anti-Money Laundering for Transactional Blockchains. *IEEE Transactions on Information Forensics and Security*, 20:391–404, 2025.

J Tang and Y. Jian. Developing an Intelligent Data Discriminating System of Anti-money Laundering based on SVM. In *Proc. IJMLC*, pages 3453–3457, Guangzhou, China, Nov. 2005.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale Information Network Embedding. In *Proc. WWW*, pages 1067–1077, 2015.

Maria Paola Tatulli, Tommaso Paladini, Mario D'Onghia, Michele Carminati, and Stefano Zanero. HAMLET: A Transformer based Approach for Money Laundering Detection. In *Proc. CSCML*, pages 234–250, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in neural information processing systems*, 30, 2017.

Chen Wang, Yuheng Qiu, Dasong Gao, and Sebastian Scherer. Lifelong Graph Learning. In *Proc. IEEE CVPR*, pages 13719–13728, New Orleans, USA, 2022a.

Cheng Wang and Hangyu Zhu. Behavior Tomographer: Identifying Hidden Cybercrimes by Behavior Interior Structure Modeling. *IEEE Transactions on Services Computing*, 2025.

Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. Streaming Graph Neural Networks via Continual Learning. In *Proc. ACM CIKM*, pages 1515–1524, 2020.

Junshan Wang, Wenhao Zhu, Guojie Song, and Liang Wang. Streaming graph neural networks with generative replay. In *Proc. ACM SIGKDD*, pages 1878–1888, Washington DC, USA, Aug. 2022b.

Su Nan Wang and Jian Gang Yang. A Money Laundering Risk Evaluation Method Based on Decision Tree. In *Proc. IEEE ICMLC*, pages 283–286, Hong Kong, China, Aug. 2007.

M. Weber, G. Domeniconi, J. Chen, Dki Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *arXiv*, 2019.

Ruofan Wu, Boqun Ma, Hong Jin, Wenlong Zhao, Weiqiang Wang, and Tianyi Zhang. GRANDE: A Neural Model over Directed Multigraphs with Application to Anti-money Laundering. In *Proc. IEEE ICDM*, pages 558–567, Orlando, FL, USA, 2022.

Sheng Xiang, Guibin Zhang, Dawei Cheng, and Ying Zhang. Enhancing Attribute-Driven Fraud Detection With Risk-Aware Graph Representation. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Y. Yang, B. Lian, L. Lian, C. Chen, and L. Pu. DBSCAN Clustering Algorithm Applied to Identify Suspicious Financial Transactions. In *Proc. IEEE CyberC*, pages 60–65, Shanghai, China, Oct. 2014.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Bad for Graph Representation? *ArXiv*, 2021.

Y. Zhang et al. Communication-Efficient Stochastic Gradient Descent Ascent with Momentum Algorithms. In *Proc. IJCAI*, pages 4602–4610, Macao, China, Aug. 2023.

Fan Zhou and Chengtai Cao. Overcoming Catastrophic Forgetting in Graph Neural Networks with Experience Replay. In *Proc. AAAI*, volume 35, pages 4714–4722, 2021.

A Limitations

A limitation of this work lies in data availability. Real-world account-level banking transaction data are highly confidential and are rarely released for public research. Therefore, we evaluate EvoDen on public Bitcoin transaction graphs and the large-scale IBM-AML benchmark. These datasets allow us to study both real blockchain fund flows and account-level AML detection under large-scale and highly imbalanced settings. However, they may still differ from real-world account-level banking transactions in data structure and behavioral patterns. In future work, we will seek collaboration with financial institutions to further validate EvoDen on real-world account-level transaction networks.

B Related Work

Continual Graph Learning Methods. Continual learning has recently been applied in dynamic graphs, dealing with the efficient update of graph models. Kou *et al.* proposed a disentangle-based continual graph representation learning method, named as DiCGRL Kou et al. [2020]. However, DiCGRL does not consolidate historical knowledge. Liu *et al.* Liu et al. [2021] proposed a topology-aware weight preserving model, named as TWP. However, regularization-based methods are difficult to obtain the optimal solution for complete data. Some research restores historical nodes in a memory buffer for continual graph representation learning Wang et al. [2020], Zhou and Cao [2021], Wang et al. [2022a]. However, these methods fail to maintain complete data distribution, so that the effect of avoiding catastrophic forgetting is limited. Therefore, Wang *et al.* proposed a streaming GNN based on generative replay, which can incrementally learn new patterns while maintaining existing knowledge without accessing historical data Wang et al. [2022b]. However, the aggregation mechanisms of graph neural networks are prone to over-smoothing as the number of layers increases, making it increasingly difficult to detect concealed money laundering activities.

C Technical Appendices

C.1 Details of the Risk-Potential Guided Walk

Given a target transaction event x_i and a sequence length l, EvoDen extracts a denoised transaction sequence L_i using the risk-potential guided walk. The walking process is described as follows.

Step 1: The walk algorithm starts from the target transaction event x_i, which is first added to the sequence L_i. If the graph is a Bitcoin transaction graph, the walk starts directly from the transaction node x_i. If the graph is an account-level transaction graph, the walk starts from the receiver account of the target transaction edge to follow subsequent fund flows.

Step 2: At each step, the walk returns to the starting position with probability α, or moves to a neighboring node with probability $1 - \alpha$. If the walk continues, the next node is sampled from the current neighborhood according to the potential energy, so that nodes with higher potential are more likely to be visited. For Bitcoin transaction graphs, the visited transaction node is added to the sequence L_i. For account-level transaction graphs, the transaction edge traversed from the current account to the selected neighboring account is added to L_i.

Step 3: Repeat step 2 until the sequence length reaches l. Here, if the walk algorithm moves to a previously visited node, it is not added again to the set L_i. To guarantee termination in small connected components, we set a maximum number of walking attempts. If fewer than l valid transaction events are collected after reaching this limit, the remaining positions are filled with a padding token and masked during sequence learning. Finally, we obtain a transaction sequence denoted as $L_i = \{x_i, x_{i1}, x_{i2}, ..., x_{i(l-1)}\}$. The feature of this transaction sequence is given by $F_i = [f_i^T, f_{i1}^T, f_{i2}^T, ..., f_{i(l-1)}^T]^T$.

C.2 Details of Continual Transformer-based Sequence Learning

Due to the powerful sequence modeling capabilities of the Transformer Vaswani et al. [2017], we input transaction sequences into a Transformer module to explore contextual associations between nodes. However, the traditional Transformer model is primarily designed for natural language processing. Hence, we improve the existing Transformer model to better suit the task of detecting

money laundering in dynamic transaction networks. In this section, we present the architecture and detailed algorithm design of the continual Transformer-based sequence learning method.

The architecture of the continual Transformer-based sequence learning model is illustrated in Fig. 3. This model primarily consists of an encoder and a decoder. The encoder is constructed from a stack of six identical layers, each containing two sub-layers. The first sub-layer employs a multi-head self-attention mechanism, and the second is a position-wise feed-forward network. Then, a residual connection is applied to each of the two sub-layers, followed by layer normalization. In this architecture, the input is a transaction sequence feature, denoted as F_i. To enable the model to adapt to varying sequence lengths, we set a maximum value ℓ for the length of the input transaction sequences. If the sampled sequence length is shorter than the maximum value, it is padded with zeros. Then, to take advantage of the sequence order, positional encodings are added to the input embeddings at the bottoms of both the encoder and decoder stacks. In the risk-potential guided sequence extraction method, the order of transactions in these sequences is determined by the transaction network structure and the importance of each neighboring node. Therefore, when applying the Transformer model to learn transaction embeddings, there is no need to design complex positional encoding methods as in Ying et al. [2021]. Instead, we utilize a simple positional encoding method, as described in the work of Vaswani et al. [2017].

Let $H_i = [h_i^T, ..., h_{i(l-1)}^T, ...]^T$ ($H_i \in \mathbb{R}^{\ell \times d}$) denote the input to the self-attention module after padding. Here, h_i is a vector obtained by concatenating the raw feature f_i of node i with its corresponding positional encoding Vaswani et al. [2017]. d denotes the dimension of all concatenated vectors. The attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is obtained as a weighted sum of the values, where the weight assigned to each value is determined by a compatibility function between the query and the corresponding key. Specifically, the input H_i is projected by three matrices $W_Q \in \mathbb{R}^{d \times d}$, $W_K \in \mathbb{R}^{d \times d}$ and $W_V \in \mathbb{R}^{d \times d}$ to the corresponding representations Q_i, K_i, V_i, having

$$Q_i = H_i W_Q, \tag{11}$$

$$K_i = H_i W_K, \tag{12}$$

$$V_i = H_i W_V, \tag{13}$$

where W_Q, W_K and W_V represent parameter matrices that need to be learned. Then, the self-attention is calculated as

$$Attention(Q_i, K_i, V_i) = softmax(\frac{Q_i K_i^T}{\sqrt{d}}) \cdot V_i, \tag{14}$$

where $\frac{Q_i K_i^T}{\sqrt{d}}$ is a matrix capturing the similarity between queries and keys. Among the 6 encoders in Fig. 3, the keys, values and queries for each encoder are derived from the output of the previous encoder. Among the 6 decoders, the queries of each decoder come from the previous decoder layer, and the keys and values come from the output of the last encoder. By utilizing these attention sub-layers, EvoDen is able to learn the association between each node and mine the hidden context information within the sequences.

Then, the output of the attention sub-layer, denoted as $Attention(Q_i, K_i, V_i)$, enters an Add&Normalize layer to prevent gradient disappearance and gradient explosion. The output of this Add&Normalize layer is represented as N_i, having

$$N_i = \mathrm{LayerNorm}(H_i + Attention(Q_i, K_i, V_i)), \tag{15}$$

where $\mathrm{LayerNorm}(\cdot)$ is a regularization function. In addition to attention sub-layers, each layer in the encoder and decoder modules includes a fully connected feed-forward network, which is applied to each position separately and identically. The fully connected feed-forward network consists of two linear transformations with a ReLU activation function applied between them. As shown in Fig. 3, when the output N_i of the Add&Normalize layer enters the feed-forward layer, the obtained output is expressed as $FFN(N_i)$, having

$$FFN(N_i) = max(0, N_i W_1 + b_1) W_2 + b_2, \tag{16}$$

where W_1 and W_2 represent two weight parameter matrices that need to be learned. b_1 and b_2 are bias vectors that need to be learned. The linear transformations are the same across different positions, but they use different parameters from layer to layer. After the feed-forward layer, $FFN(N_i)$ enters an Add&Normalize layer again. By stacking 6 encoder modules, transaction embeddings corresponding to all transactions in the sequence L_i can be obtained, denoted by $\widetilde{F}_i$.

Table 5: Statistics of the IBM-AML datasets

Dataset	# Nodes	# Edges	Illicit Rate	Time Span
AML Small-HI	515,088	5,078,345	0.102%	10 days
AML Small-LI	705,907	6,924,049	0.051%	10 days
AML Medium-HI	2,077,023	31,898,238	0.110%	16 days
AML Medium-LI	2,032,095	31,251,483	0.051%	16 days
AML Large-HI	2,116,168	179,702,229	0.124%	97 days
AML Large-LI	2,070,980	176,066,557	0.057%	97 days

D Experimental Details

D.1 Details of Datasets

Due to the high confidentiality of data in banks, we conduct experiments on two real-world bitcoin transaction datasets and a synthetic transaction dataset provided by IBM.

- Elliptic Weber et al. [2019]: This is currently the largest public dataset of bitcoin transactions. This dataset consists of 203,769 transaction nodes and 234,355 directed fund transfer edges, each representing the movement of monetary value from one node to another. 4,545 nodes are labeled as illegal. 42,019 are labeled as legal, and the remaining transactions are unlabeled. These node entities consist of 166 features. The label is set to 1 for illicit nodes and 0 for normal nodes. This dataset contains 49 time steps, forming a dynamic transaction network. A whole transaction network refers to the integration of data from all 49 time steps into a unified structure.
- WalletExplorer[4]: WalletExplorer is a smart bitcoin block explorer. We collect a part of bitcoin transaction data from March 2014 to May 2021, which includes 62,217 money laundering transactions and 255,770 normal transactions. We label money laundering transactions as 1 and normal transactions as 0. Here, labels of money laundering transactions is provided by the *BitLaunder* service and *BitcoinFog* service. Inspired by Hu et al. Hu, Yining and Seneviratne, Suranga and Thilakarathna, Kanchana and Fukuda, Kensuke and Seneviratne, Aruna [2019], we construct 14 features for each transaction. These include in-degree, out-degree, the sum/mean/standard deviation of output and input amounts, the ratio of total input to total output, the ratio of in-degree to out-degree, betweenness centrality, clustering coefficient, PageRank score, and the size of the weakly connected component containing the transaction. Then, we divide the data set into half a month as a time step to form a dynamic transaction network.
- IBM-AML Altman et al. [2023]: To assess the robustness of our method under different data scales and highly imbalanced illicit ratios, we conduct experiments on the IBM Transactions for AML benchmark generated by the AMLworld agent-based simulator. The released benchmark consists of six datasets grouped into two categories according to illicit transaction prevalence: HI (Higher Illicit) and LI (Lower Illicit). Each category further provides three scales, namely Small, Medium, and Large, resulting in a comprehensive suite covering a wide range of difficulty and data volume. The detailed statistics including the number of accounts, transaction counts, days spanned, and laundering transaction volume for each split are summarized in Table 5.

D.2 Details of Baseline Methods and Implementation Settings

To highlight the effectiveness of our proposed method, we compare EvoDen with 15 baseline methods. We first compare with three widely used classical machine learning methods and five static graph learning methods to analyze the representation learning capabilities of EvoDen on the whole transaction network.

[4]https://www.walletexplorer.com

- Logistic Regression (LR) Hosmer and Lemeshow [2000], Random Forest (RF) Breiman [2001] and Extreme Gradient Boosting (XGB) Chen and Guestrin [2016]: These are traditional machine learning algorithms widely used for classification tasks.
- LINE Tang et al. [2015]: This is a static graph embedding model for large-scale networks.
- GCN Alarab et al. [2020]: This is a classic work that applies Relational-GCNs to AML by learning node representations from the transaction graphs for illicit activity classification.
- GraphSAGE Hamilton et al. [2017]: GraphSAGE is widely recognized as classical algorithms with excellent performance for learning graph embeddings.
- Inspection-L Lo et al. [2023]: This is the latest self-supervised graph learning method for AML.
- FraudGT Lin et al. [2024b]: It is a graph Transformer-based fraud detection model that incorporates edge-aware message passing and attention biases to improve efficiency and effectiveness on large-scale financial transaction graphs.

Then, we evaluate the performance of EvoDen on dynamic transaction networks. The comparison methods are as follows.

- DNE Du et al. [2018]: This is an extension of LINE to update node representations efficiently in a dynamic environment.
- SingleGNN: It is a GraphSAGE trained individually at each time step.
- OnlineGNN: This is a variant of GraphSAGE trained in an online manner but without any consideration for knowledge consolidation.
- GNN-EWC and GNN-ER: The two methods are both based on OnlineGNN and directly adopt continual learning approaches: EWC Kirkpatrick et al. [2017] and ER Rolnick et al. [2019] respectively.
- GRANDE Wu et al. [2022]: This is a graph representation learning framework for AML on directed multigraphs via efficiently incorporating directional information.
- HAMLET Tatulli et al. [2023]: This is a hierarchical Transformer-based AML model that leverages transaction-level and sequence-level self-attention to detect complex money laundering patterns from financial transaction sequences.

For implementation, our hardware system includes a server containing one CPU with an Intel Xeon Gold 5220, two Tesla V100 GPUs, and 32GB of memory. The software environment is based on Python 3.9.

D.3 Implementation Details

Static transaction graph setting. In the static setting, we evaluate the representation learning ability of EvoDen on the whole transaction graph. For the Elliptic dataset, we follow the chronological order of the 49 timesteps and use the first 34 timesteps for training and the remaining 15 timesteps for testing, corresponding to an approximately 7:3 split by time. The same chronological splitting principle is applied to WalletExplorer to avoid temporal leakage. In this setting, EvoDen is trained without replay, so the evaluation mainly reflects the effectiveness of the risk-potential guided sequence extraction and the reconstruction-based sequence representation learning module.

For all compared methods in the static setting, the batch size is set to 512 and the number of training epochs is set to 1000. For LINE, we use both first-order and second-order proximity representations and concatenate them as the final embedding. For GCN and GraphSAGE, we use two hidden layers with output dimensions 128 and 64, respectively. For EvoDen, transaction sequences are first extracted by the risk-potential guided walk, and transaction representations are then learned through the reconstruction-based Transformer module. The learned embeddings are fed into the downstream classifier for money laundering detection.

Dynamic transaction graph setting. In the dynamic setting, models are evaluated under a streaming transaction scenario, where newly observed transactions arrive over time and the model is updated incrementally. The samples at the current timestep are split into training and testing sets with a 7:3

Table 6: Performance results for money laundering transaction detection on the Elliptic dataset. Best and second results are highlighted by **bold** and underline.

Method	Elliptic				
	Precision ↑	Recall ↑	F1 ↑	AUC ↑	MCC ↑
LR	0.3193±0.0000	0.7193±0.0000	0.4422±0.0000	0.8983±0.0000	0.4272±0.0000
RF	0.9554±0.0088	0.7211±0.0028	0.8219±0.0039	0.9195±0.0041	0.8204±0.0046
XGB	0.9085±0.0000	0.7239±0.0000	0.8058±0.0000	0.9211±0.0000	0.7997±0.0000
LINE	0.7562±0.0081	0.6032±0.0134	0.6711±0.0079	0.7487±0.0019	0.6553±0.0088
GCN	0.8992±0.0021	0.6978±0.0096	0.7858±0.0055	0.8201±0.0014	0.7799±0.0057
GraphSAGE	0.9032±0.0023	0.8637±0.0082	0.8830±0.0038	0.9095±0.0071	0.8752±0.0040
Inspection-L	0.8829±0.0134	0.7012±0.0078	0.7816±0.0095	0.8667±0.0040	0.7734±0.0104
FraudGT	0.9074±0.0071	0.8639±0.0059	0.8852±0.0066	0.9289±0.0036	0.8641±0.0075
EvoDen (Static)	**0.9699*****±0.0073**	**0.9019******±0.0044**	**0.9347******±0.0050**	**0.9371******±0.0023**	**0.9306******±0.0068**
DNE	0.7029±0.0136	0.5907±0.0055	0.6419±0.0115	0.6762±0.0178	0.6133±0.0089
SingleGNN	0.7421±0.0082	0.6129±0.0119	0.6713±0.0094	0.7239±0.0065	0.6551±0.0073
OnlineGNN	0.8270±0.0090	0.6713±0.0104	0.7411±0.0098	0.7992±0.0045	0.7393±0.0070
GNN-EWC	0.8616±0.0051	0.7791±0.0081	0.8183±0.0074	0.8334±0.0077	0.8120±0.0069
GNN-ER	0.8739±0.0013	0.7512±0.0089	0.8079±0.0057	0.8209±0.0079	0.7822±0.0084
GRANDE	0.8205±0.0010	0.6329±0.0021	0.7146±0.0016	0.7896±0.0055	0.7031±0.0013
HAMLET	0.6378±0.0082	0.5903±0.0023	0.6131±0.0052	0.6908±0.0058	0.5799±0.0070
EvoDen (Dynamic)	**0.9391******±0.0043**	**0.8779******±0.0077**	**0.9075******±0.0055**	**0.9126******±0.0024**	**0.8902******±0.0033**

ratio. This split is performed strictly within the current timestep, ensuring that no future transaction information is introduced into either training or evaluation.

For dynamic graph and continual learning baselines, including OnlineGNN, GNN-EWC, GNN-ER, and EvoDen, the model parameters from the previous timestep are loaded and updated using the current incremental data. For replay-based methods such as GNN-ER and EvoDen, we replay 300 historical samples at each new timestep. The batch size is set to 128, and each model is trained for 100 epochs at each timestep. The restart probability of the risk-potential guided walk is set to $\alpha = 0.15$.

D.4 Complete Performance Results for Money Laundering Transaction Detection on the Elliptic and WalletExplorer Datasets

This section provides the complete detection results on Elliptic and WalletExplorer under both static and dynamic transaction graph settings. In the main paper, we report the most informative metrics for imbalanced AML detection, including F1-score, AUC, and MCC. Here, we further provide Precision and Recall, as shown in Table 6 and 7. It is clear that EvoDen significantly outperforms other continual learning methods in terms of precision, recall, F1-score, AUC and MCC. Among these, DNE does not utilize transaction features, resulting in poor performance. SingleGNN retrains the model with incremental data at each new step, and OnlineGNN updates the model solely with incremental data. These approaches lead to catastrophic forgetting. GNN-EWC and GNN-ER improve performance compared to SingleGNN and OnlineGNN, because GNN-EWC and GNN-ER consolidate old knowledge through continual learning techniques. However, GNN-ER replays historical data based on experience, so it fails to maintain the complete historical knowledge. Besides, the ability of GNN-EWC to preserve historical knowledge is also insufficient. In EvoDen, we employ the Transformer to learn the local data distribution and use a clustering-based generative replay method to preserve the global data distribution. This approach significantly enhances the performance of money laundering detection in dynamic transaction networks. Similarly, the GRANDE method, which is based on enhanced graph learning, and the HAMLET method, which relies solely on the Transformer, exhibit inferior performance compared to EvoDen. This is attributed to their incapacity to effectively tackle challenges related to concealed money laundering activities and catastrophic forgetting issues.

To provide a more intuitive interpretation of the reported metrics, we present the confusion matrices of baseline methods and EvoDen on the Elliptic dataset. As shown in Fig. 6, EvoDen yields a

Table 7: Performance results for money laundering transaction detection on the WalletExplorer dataset. Best and second results are highlighted by **bold** and underline.

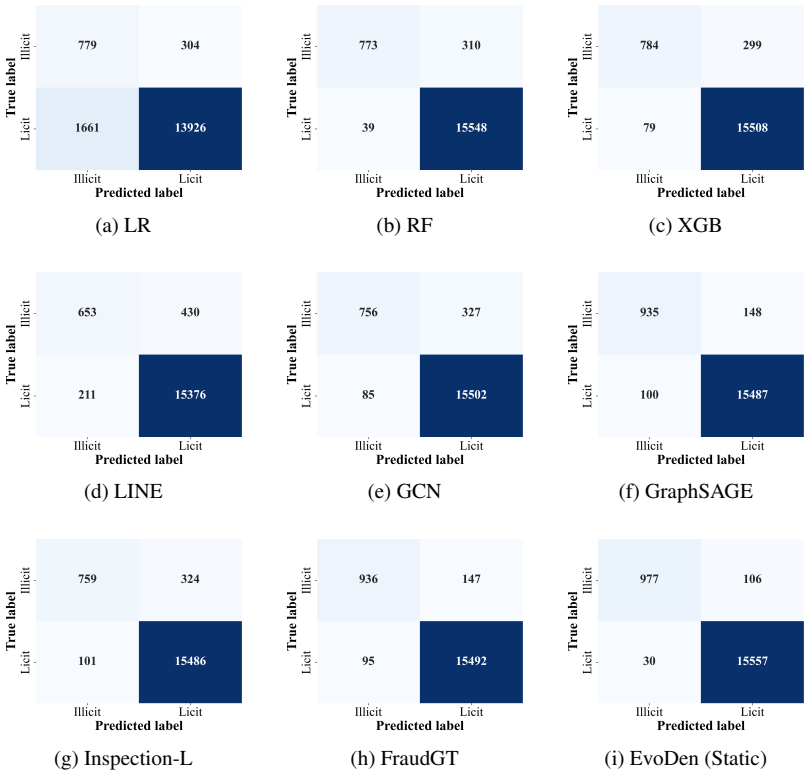
Method	WalletExplorer				
	Precision ↑	Recall ↑	F1 ↑	AUC ↑	MCC ↑
LR	0.6358±0.0000	0.5147±0.0000	0.5689±0.0000	0.7554±0.0000	0.4852±0.0000
RF	0.8738±0.0054	0.7591±0.0031	0.8124±0.0079	0.8429±0.0089	0.7753±0.0111
XGB	0.8209±0.0000	0.7745±0.0000	0.7970±0.0000	0.8343±0.0000	0.7520±0.0000
LINE	0.6326±0.0093	0.6619±0.0157	0.6469±0.0131	0.8003±0.0040	0.5627±0.0208
GCN	0.8262±0.0122	0.7320±0.0092	0.7763±0.0064	0.8036±0.0059	0.7300±0.0124
GraphSAGE	0.8647±0.0051	0.8465±0.0108	0.8555±0.0089	0.8807±0.0035	0.8224±0.0176
Inspection-L	0.8358±0.0009	0.7747±0.0045	0.8041±0.0037	0.8203±0.0113	0.7614±0.0082
FraudGT	0.8720±0.0083	0.8516±0.0093	0.8617±0.0071	0.8842±0.0022	0.8300±0.0086
EvoDen (Static)	**0.9028******±0.0086**	**0.8750******±0.0133**	**0.8887******±0.0091**	**0.9092******±0.0058**	**0.8634******±0.0125**
DNE	0.6723±0.0028	0.6018±0.0104	0.6351±0.0057	0.6030±0.0033	0.6310±0.0040
SingleGNN	0.7662±0.0089	0.6219±0.0055	0.6865±0.0064	0.7332±0.0021	0.6649±0.0097
OnlineGNN	0.7823±0.0114	0.6831±0.0077	0.7293±0.0082	0.7698±0.0045	0.7122±0.0050
GNN-EWC	0.8230±0.0030	0.7859±0.0096	0.8040±0.0088	0.8175±0.0062	0.7816±0.0143
GNN-ER	0.8551±0.0071	0.8023±0.0139	0.8279±0.0097	0.8326±0.0046	0.7930±0.0081
GRANDE	0.7411±0.0055	0.6524±0.0124	0.6939±0.0081	0.8015±0.0060	0.6911±0.0028
HAMLET	0.6377±0.0138	0.5631±0.0067	0.5980±0.0094	0.7018±0.0111	0.5778±0.0084
EvoDen (Dynamic)	**0.8874******±0.0067**	**0.8510******±0.0029**	**0.8688******±0.0071**	**0.8536******±0.0052**	**0.8936******±0.0070**

substantially lower false negative rate while maintaining a low false positive rate, which is critical in real-world money laundering detection scenarios. This indicates that EvoDen can effectively denoise complex transaction networks and identify crucial money laundering clues, thereby improving the efficiency of money laundering detection. In practical deployment, suspicious transactions flagged by EvoDen should be treated as risk alerts rather than final decisions, and human experts should review these alerts before any compliance or enforcement action is taken.

D.5 Additional Analysis on IBM-AML

We further evaluate our method on the IBM-AML benchmark from two complementary perspectives: 1) scalability on substantially larger transaction graphs and 2) practical utility measured in monetary terms. Specifically, IBM-AML provides large-scale synthetic transaction networks with different graph sizes (Small/ Medium/ Large) and illicit prevalence regimes (HI/ LI), which enables systematic evaluation under conditions where illicit behaviors are highly imbalanced and may become harder to distinguish as graph size increases. The experimental setups for all evaluated models on the IBM-AML benchmark are kept consistent with those in FraudGT Lin et al. [2024b] for consistency and reproducibility. Following common practice for imbalanced AML detection, we report F1-score as the primary effectiveness metric. In addition, to quantify economic value beyond statistical performance, we also report the Recovered Laundering Amount Ratio (RLAR) on the test set, defined as the total laundering monetary volume associated with correctly detected illicit activities divided by the total laundering volume in the test set. RLAR directly reflects the fraction of illicit funds that could be intercepted by the model in practice, which is particularly relevant in AML tasks where maximizing recovered illicit volume is often a primary operational objective.

Table 2 reports the detection performance (F1) and the economic benefit (RLAR) of different methods on the IBM-AML benchmark. Traditional classifiers (LR/RF/XGB) are largely ineffective in this highly imbalanced setting, producing near-zero F1 and negligible RLAR across all configurations. Static graph representation baselines (e.g., LINE, GCN, Inspection-L) provide noticeable improvements but remain limited, particularly under low-illicit regimes where illicit signals are weaker and more easily diluted by dominant benign behaviors. In contrast, FraudGT improves both F1 and RLAR by more effectively exploiting edge information and explicitly accommodating directed multigraph structures. More importantly, EvoDen exhibits strong and robust performance by directly targeting these concealment mechanisms through spatio-temporal dual denoising. Under evaluation on the whole transaction network, EvoDen achieves near-best performance in terms of both F1 and RLAR across different IBM-AML configurations. Notably, even in the most challenging Large-LI setting,



(a) LR (b) RF (c) XGB

(d) LINE (e) GCN (f) GraphSAGE

(g) Inspection-L (h) FraudGT (i) EvoDen (Static)

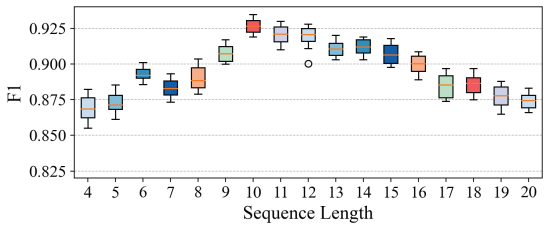
Figure 6: Confusion matrices of different methods on Elliptic dataset.

where its F1 is slightly lower than FraudGT, EvoDen attains a higher RLAR. This indicates that the proposed sequence extraction more effectively surfaces laundering-relevant fund-flow signals, thereby enabling a larger fraction of illicit funds to be captured in practice. In the dynamic setting, EvoDen maintains consistently strong performance as new transactions stream in. This robustness stems from our continual Transformer-based sequence learning framework, which incrementally integrates new observations while preserving historical laundering cues that are crucial for early anomaly detection in evolving networks. Moreover, the clustering-based replay selects diverse and discriminative historical samples in the learned embedding space, mitigating bias toward recent or frequent patterns and preventing representation drift toward benign-dominated behaviors, thereby sustaining detection performance over long-term dynamic transaction networks.

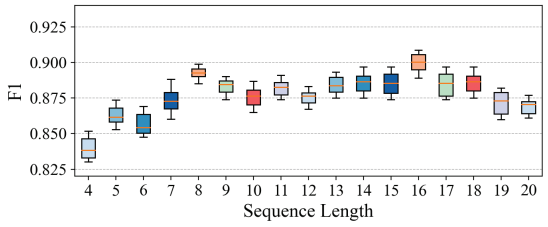
D.6 Parameter Sensitivity Analyses

In this section, we analyze the sensitivity of our model to three factors that may influence its detection performance. The first is the sequence length l, which controls the amount of fund-flow context extracted by the risk-potential guided walk. The second concerns the expenditure-to-income (E/I) ratio, which is used as a behavioral component in the node potential function and may approach a steady-state value over long time horizons. The third factor is the learning rate of the classifier, which affects the optimization dynamics during training. By examining these factors, we aim to assess the robustness of the proposed method with respect to feature design choices and training configurations.

1) The Effect of Sequence Length. The sequence length l determines how much neighborhood context is retained in the extracted transaction sequence. A short sequence may fail to capture sufficient fund-flow information, while an excessively long sequence may introduce noisy or redundant transactions and increase computational cost. As shown in Fig. 7, the F1-score generally increases first and then decreases as l grows. On Elliptic, EvoDen reaches the best performance around $l = 10$, indicating that a moderate sequence length is sufficient to capture useful laundering-related context. On WalletExplorer, the model also achieves strong performance at moderate sequence lengths, while longer sequences introduce higher sampling and Transformer computation cost. Therefore, we set



(a) Elliptic



(b) WalletExplorer

Figure 7: Performance results under different sequence length.

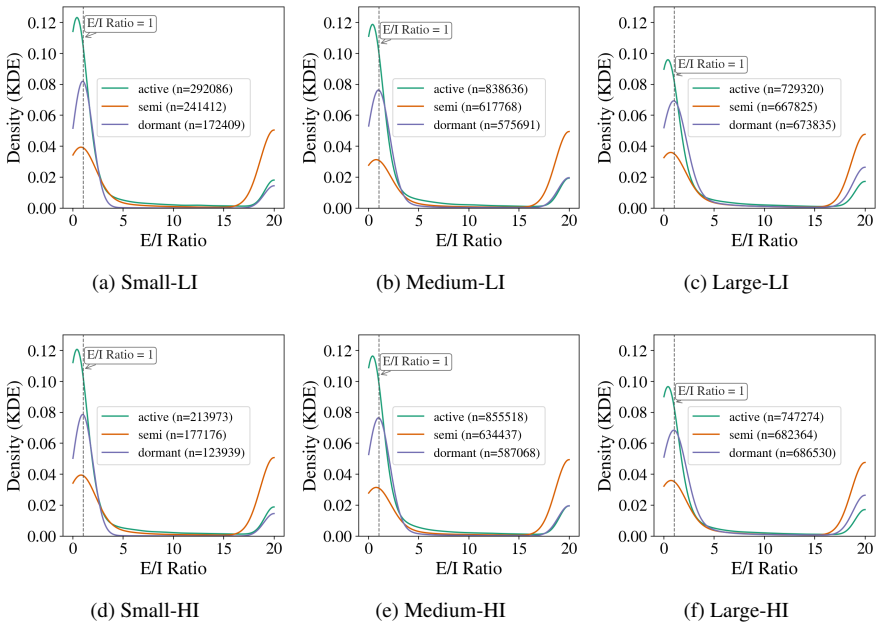
Table 8: Ablation on the expenditure-to-income ratio on IBM-AML in terms of F1-score.

Datasets \ Methods	EvoDen (Static)	EvoDen (Static) with E/I = 1	EvoDen (Dynamic)	EvoDen (Dynamic) with E/I = 1
AML Small-HI	**0.7984**	0.7802	**0.6518**	0.6423
AML Small-LI	**0.4918**	0.4711	**0.4003**	0.3858
AML Medium-HI	**0.7850**	0.7789	**0.6241**	0.5990
AML Medium-LI	**0.4521**	0.4366	**0.3655**	0.3544
AML Large-HI	**0.7597**	0.7515	**0.6512**	0.6361
AML Large-LI	**0.3601**	0.3428	**0.3101**	0.3029

$l = 10$ for Elliptic and $l = 8$ for WalletExplorer in the main experiments to balance detection performance and computational efficiency.

2) The Effect of E/I Ratios. The E/I ratio is a component of our node potential function, intended to reflect the balance tendency of an account within an evolving transaction network. A natural concern is that over a sufficiently long horizon, many accounts may approach a near steady-state with $\mathrm{E}/\mathrm{I} \approx 1$, which could reduce the discriminative power of this feature. However, our method is designed for continuously evolving transaction networks, where the E/I ratio is computed as a time-varying signal from the running transaction history up to the current time. In practical financial systems, account behaviors typically undergo multiple phases (e.g., accumulation, circulation, transfer, and inactivity), implying that many accounts are not necessarily in a steady regime with $\mathrm{E}/\mathrm{I} \approx 1$ at a given time. To empirically examine this issue, we first analyze the distribution of E/I ratios across accounts over the entire observation window on the IBM-AML dataset. Specifically, we define an account's activity score as the ratio of active days to total days, and partition accounts into dormant, semi-active, and active groups using two thresholds (by default, the lower/upper tertiles). For each group, we estimate the distribution of E/I ratios using kernel density estimation, where each group's contribution is weighted by its prevalence. Fig. 8 shows that the observed E/I ratio is not concentrated solely around 1. The active and semi-active groups exhibit substantial dispersion and pronounced mass away from 1, indicating that non-steady intermediate states remain prevalent in large-scale AML graphs. This supports the use of E/I ratio as a meaningful component in our node potential function.

To further clarify the role of E/I ratio, we conduct an ablation study on IBM-AML in which all other components of the node potential function remain unchanged and the E/I ratio of all accounts is replaced with a constant value of 1. We compare the resulting performance against the original setting. As shown in Table 8, fixing the E/I ratio to 1 leads to a consistent but moderate decrease in



(a) Small-LI (b) Medium-LI (c) Large-LI

(d) Small-HI (e) Medium-HI (f) Large-HI

Figure 8: E/I Ratio distribution on IBM-AML.

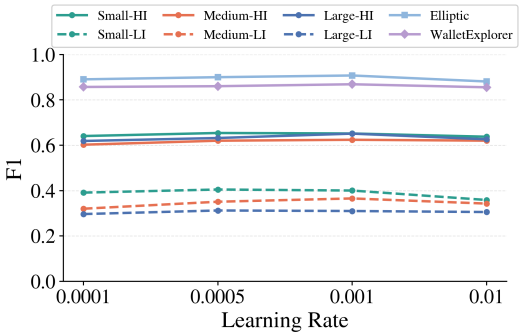


Figure 9: The impact of learning rate on the performance of EvoDen in terms of F1-score.

F1 across all IBM-AML settings for both static and dynamic variants of EvoDen. The performance drop is generally limited, indicating that while the E/I ratio provides useful behavioral information, the model does not overly rely on this feature. Notably, EvoDen retains strong detection performance after the ablation, suggesting that other components of the node potential function, such as transaction frequency and average transaction amount, remain effective in capturing anomalous account behavior.

3) The Effect of Learning Rate. We investigate the effect of the learning rate on model performance under the dynamic setting across eight datasets, as shown in Fig. 9. We vary the learning rate in $\{0.0001, 0.0005, 0.001, 0.01\}$. It can be observed that our model exhibits high stability and robustness. Specifically, the F1-scores remain relatively constant across a wide range of learning rates, indicating that the model is not sensitive to this hyperparameter. When moving outside this stable regime, a very small learning rate may slightly reduce F1 due to slower convergence under a fixed training budget, whereas an overly large learning rate can cause mild degradation due to less stable updates. Therefore, we set the learning rate to 0.001.

D.7 Time Performance Evaluation under Different Data Increment Scales

To evaluate the scalability of our EvoDen under different data increment scales, we evaluate the processing capability of EvoDen when facing a sudden surge in data volume within a short period.

Table 9: Processing Time at Different Increments of Data

△Num	5k	10k	50k	100k	150k	200k
Time on Elliptic (s)	1.24	2.63	8.55	18.27	31.30	44.32
Time on WalletExplorer (s)	0.76	1.70	7.01	15.13	25.39	32.85

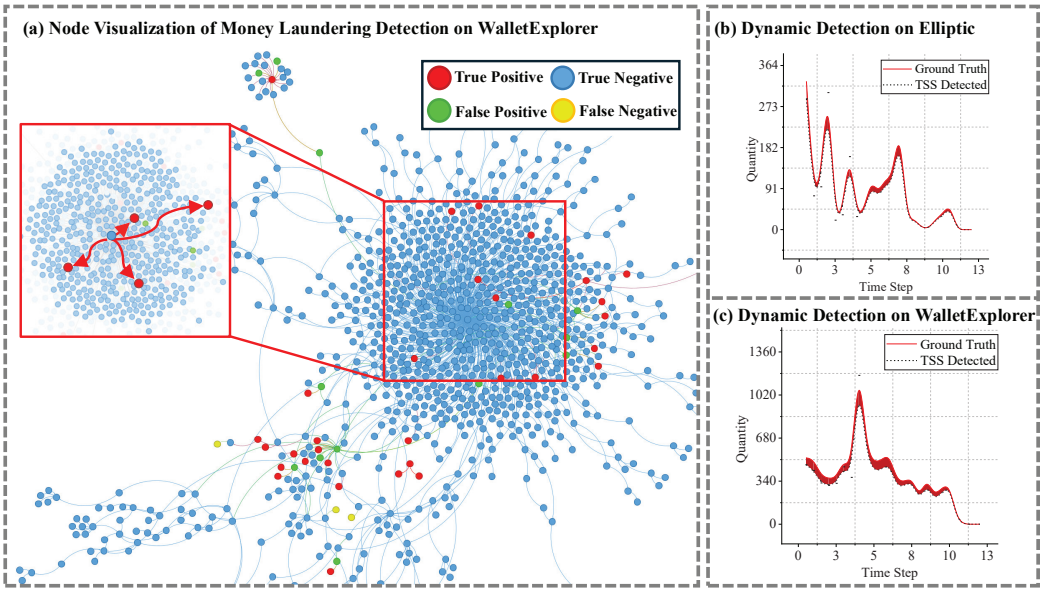


Figure 10: Detection result visualization of EvoDen.

In the experiment, we measure the processing time required by EvoDen as the data incrementally varied from 5, 000 to 200, 000. Table 9 summarizes the total processing time required for each data increment on two datasets. We observe that the runtime increases with the increment scale but remains efficient on both datasets. For Elliptic, the processing time increases from 1.24s at 5k to 18.27s at 100k, and reaches 44.32s at 200k. For WalletExplorer, it rises from 0.76s at 5k to 15.13s at 100k, and reaches 32.85s at 200k. These results indicate that EvoDen can complete each incremental update within tens of seconds even at large increment scales, demonstrating good scalability. In real-world deployment, further acceleration can be achieved via parallelization or partition-based processing to improve throughput under extreme data influx.

D.8 Case Study

To intuitively reflect the effectiveness of our EvoDen, we visualize the detection results in Fig. 10. In Fig. 10(a), we illustrate the transaction detection results of EvoDen on a subset of the WalletExplorer test set. The magnified red box highlights that EvoDen successfully detects all concealed money laundering nodes within a dense cluster of normal nodes. This demonstrates the significant denoising effect of our proposed EvoDen, which can isolate weak but important laundering signals embedded in complex transaction networks. In Fig. 10(b) and 10(c), we show the detection results of EvoDen as the transaction network evolves dynamically. The red line represents the number of labeled money laundering nodes at each timestep, while the black line indicates the number of money laundering nodes correctly detected by EvoDen. The red-shaded area indicates instances where money laundering nodes were missed. As observed, the two lines exhibit minimal deviation, demonstrating that EvoDen maintains stable performance despite the dynamic changes in transaction networks. This stability confirms the strength of our continual Transformer module, which incrementally learns from new sequences while retaining critical historical knowledge via reconstruction-based and clustering-guided generative replay. Notably, even when the number of money laundering nodes exhibit sudden changes in certain timesteps (e.g., spikes in red line height), EvoDen rapidly adapts without significant performance degradation. These results suggest that EvoDen not only captures evolving laundering patterns, but also exhibits strong resilience to spatio-

temporal noise and knowledge forgetting. This enhances the ability of the financial system to control money laundering risks in the face of concealment tactics employed by criminals.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are empirically verified in Section 5, in particular Table 1 and 2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We mention our limitations in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not present formal theoretical results, theorems, or proofs. The proposed method is primarily algorithmic and empirically evaluated.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose all information needed for reproduction in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide code for the final paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Each experiment is repeated 10 times, and the paper reports mean performance with standard deviation in Table 1. The paper also conducts paired t-tests against the second-best method and marks statistically significant improvements in Table 1 and 2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information about the computer resources used in Appendix D.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and understood the code of ethics and have made every effort to adhere to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts in supporting financial institutions and regulators in detecting concealed laundering activities. It also notes deployment-related risks, such as a small number of false positives, and emphasizes that practical use should involve human review and responsible auditing in Appendix D.4.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, pre-trained language models, image generators, or unsafe scraped datasets. The proposed model is designed for AML detection and is evaluated on public or benchmark transaction datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available Elliptic and IBM-AML datasets, both of which are properly cited in the paper. We adhere to the licenses and terms of use as specified by their original creators.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release our codebase along with detailed README files. The release does not include source code copied from external libraries; all code provided is our own.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not use crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not use crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used an LLM solely for editing assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.