Diga: Guided Diffusion Model for Graph Recovery in Anti-Money Laundering

Xujia Li*
Hong Kong University of Science and Technology
Hong Kong SAR, China
xligm@connect.ust.hk

Yuan Li
WeBank
Shenzhen, China
euleli@webank.com

Xueying Mo
WeBank
Shenzhen, China
helenmo@webank.com

Hebing Xiao
WeBank
Shenzhen, China
hebingxiao@webank.com

Yanyan Shen
Shanghai Jiao Tong University
Shanghai, China
shenyy@sjtu.edu.cn

Lei Chen†
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
leichen@cse.ust.hk

ABSTRACT

With the upsurge of online banking, mobile payment, and virtual currency, new money-laundering crimes easily conceal in the enormous transaction volume. The traditional rule-based methods with large amounts of alerting thresholds are already incapable of handling the fast-changing transaction networks. Recently, the DL models represented by the graph neural networks (GNNs) show the potential to capture money-laundering modes with high accuracy. However, most related works are still far from practical deployment in the industry. Based on our practice at WeBank, there are three major challenges: Firstly, supervised learning is infeasible facing the extraordinarily large-scale but imbalanced data, with hundreds of millions of active accounts but only thousands of anomalies. Secondly, the real-world transactions form a sparse network with millions of isolated user groups, which overflows the expressive ability of current node-level GNNs. Thirdly, the explanation for each suspicious account is mandatory by the government for double check, which conflicts with the black-box nature of most DL models. Therefore, we proposed Diga, the first work to apply the diffusion probabilistic model to a graph anomaly detection problem with three novel techniques: the biased K-hop PageRank, the semi-supervised guided diffusion and the novel weight-sharing GNN layer. The effectiveness and efficiency of Diga are verified via intensive experiments on both industrial and public datasets.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Anomaly detection**.

*Corresponding author.
†Also with Hong Kong University of Science and Technology, Hong Kong SAR, China.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599806

KEYWORDS

Anti-Money Laundering; Graph Anomaly Detection; Diffusion Model

ACM Reference Format:
Xujia Li, Yuan Li, Xueying Mo, Hebing Xiao, Yanyan Shen, and Lei Chen. 2023. Diga: Guided Diffusion Model for Graph Recovery in Anti-Money Laundering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599806

1 INTRODUCTION

Money laundering became one of the major threats to national public security [18]. According to the SIPRI institute, crime groups laundered about 1.6 trillion dollars in 2022, which is about 2.7% of the global GDP [12]. Thus, Countries worldwide keep strengthening Anti-Money Laundering (AML) supervision and have issued huge fines to AML obligation entities due to their poor AML control [17]. In China, there were about 370 million in administrative penalties to commercial banks, a yearly increase of 30% in 2022. The tremendous scale of money laundering and the corresponding penalties indicate that the banking industry still lacks effective technology to detect suspicious accounts in transaction networks [3].

Existing AML systems can be categorized into rule-based and data-mining-based methods. Rule-based detection methods are still the mainstream in the deployed AML systems [25]. Based on expert knowledge, banks have developed many indicators and alerting rules, such as the frequency of large transactions (over $100 thousand) with an account located in high-risk areas at midnight. Once some accounts exceed the preset thresholds, they will be identified as suspects. Although rule-based methods are highly logical and expediently integrated into the automated decision process, the heavy labor participation is inadequate in facing the exploding data volume [18]. Meanwhile, the predefined rules can be easily attacked and circumvented by unknown money-laundering patterns [2]. Therefore, researchers have investigated the feasibility of data mining techniques like machine learning (ML) and deep learning (DL) in AML scenarios. Support vector machines (SVM) project high-dimensional features of transactions to a hyperplane and detect anomalies by learning decision boundaries [29]. Decision-tree-based models are also frequently mentioned because

they can provide feature importance analysis as explanations [13]. DL techniques like AutoEncoder and recurrent neural networks (RNN) are proven effective in AML via various embedding methods [17]. Notably, with the booming of GNNs, considering the topological structure and neighboring information further facilitates the representation of transaction networks [26]. However, most related works are still stuck in the research stage and far from practical deployment in real industry [14]. We conclude three major challenges based on our practice at WeBank, the first internet bank in China:

The sparsity of networks: in industry, one specific bank can only monitor internal transactions because of the privacy requirement. The cross-bank transactions are untraceable and break the entire transaction chains, constituting a sparse and fragmented transaction graph with millions of isolated user groups, as shown in the left part of Figure 1. This characteristic degrades the embedding quality of the methods like GNNs and RNNs, which prefer continuous and regular data formats [44].

The massive and unbalanced data: the number of daily transactions is over 100 million orders of magnitude in WeBank, and the number of active user accounts exceeds 200 million. The data volume overflows the expressive ability of most DL models, and the training duration is unacceptable if directly operating on the completed network. Moreover, less than 0.02% accounts are manually labeled as suspicious accounts. The unbalanced distribution causes a low detection rate of most supervised-learning models, and the new money-laundering approach can easily escape their trained consciousness. In addition, the existing GNN-based models mainly focus on node-level anomaly detection [23], which is infeasible to embed node by node with such data volume.

The model interpretability: although the black-box models with neural networks show strength in the detection accuracy over the white-box models such as decision trees, most banks cannot directly report the network outputs, i.e., the probabilities, to the government regulators for double check [17]. The rationale behind each suspicious case should be clarified for final conviction. Providing human interpretable explanations becomes the major obstacle for the DL methods being adopted by AML obligation entities.

To address the above intractable AML challenges in the banking industry, we innovatively introduce a **semi-supervised subgraph recovery approach** to the AML problem. Recovery means reconstructing the original subgraph to an anomaly-free subgraph. Each original subgraph is extracted by a biased K-hop PageRank from the whole transaction network. We solve the AML problem on the subgraph level out of three considerations: Firstly, the nature of collaborative crime appears as the anomaly node groups in a local subgraph rather than a single node or single transaction. Secondly, the user groups naturally appear as isolated but inside-dense subgraphs, which resolves the mentioned sparsity problem. Thirdly, the vast transaction graph makes it difficult for GNNs to directly embed every single node into a unified latent space. The reasons for utilizing semi-supervised learning lay in two aspects: On the one hand, the pure unsupervised reconstruction approach assumes that the abnormal information can hardly be reconstructed after an encoder-decoder process, which has been proven in anomaly recognition for image, molecule, and social network data [42, 43]. They can inductively catch those unseen anomalies by learning on large amounts of normal data [14], which is suitable to counter the challenge caused by the massive and unbalanced transaction graph. On the other hand, the limited labeled samples condense valuable expert knowledge about AML, which guarantees the detection precision against observed money-laundering patterns in a supervised manner [6]. Therefore, we take advantage of both ideas with a semi-supervised guided diffusion by magnifying the reconstructed error with the usage of labeled anomaly samples.

Following the above insights, we present **Diga**: a Guided **Di**ffusion Model for **G**raph Recovery in **A**nti-Money-Laundering, which is an end-to-end framework from the subgraph extraction to the evaluation and explanation of each suspicious account. The framework can be divided into the following steps:

Firstly, **the subgraph sampler** extracts the most-informative neighbors of each user account and constructs an adaptive subgraph surrounding it. We can treat each subgraph as an independent training sample and solve the problem on the subgraph level, making it possible to utilize GNN and pooling method on a small subgraph rather than directly operating on the large-scale dataset. Secondly, these subgraphs are treated as original graphs being fed into **the guided diffusion module**, which is intended to recover an arbitrary subgraph to an anomaly-free version. Inspired by the recent breakthrough of generative models in computer vision, the probabilistic diffusion models outperform the generative adversarial networks (GANs) in image generation on various aspects, such as the conditional synthesis, training stability, and scalability on larger datasets [4, 10, 30, 37]. Considering these advantages are highly in accord with the demands in AML scenarios, we are first to invoke the diffusion model as the base model in a graph anomaly detection problem and verify its practicality. The diffusion module injects a series of noise from a normal distribution iteratively onto the original subgraph in the forward diffusion process. Then in the reversed denoising processing, the denoising GNN network predicts the injected noise to reconstruct the graph. In addition, a novel graph classifier is designed to guide the recovering direction towards a "healthy" subgraph by revising the predicted noise in the denoising process. The guiding classifier ensures that only suspicious regions are changed while the rest of the subgraph remains unchanged. Eventually, in AML detecting, the reconstruction error between the original subgraph and the recovered version measures the anomaly score of each account. The divergence in node attributes can be visualized with an anomaly heat map, which provides an adequate explanation for each suspicious account, e.g., the reconstruction error of the transaction frequency on the weekend is much larger than other benign accounts. Furthermore, we design **the weight-sharing GNN layer** with an alternating training pipeline, which highly resolves the current drawback of training efficiency in the diffusion model with two neural networks for denoising and classifying [39]. Our main contributions are summarized as follows:

- We novelly design the semi-supervised subgraph recovery approach for the accurate and explainable AML prediction.
- We propose three novel methods to support Diga in the large and sparse real-world transaction networks: the biased PageRank for high-quality subgraphs, the guided graph diffusion for anomaly detection, and the weights-sharing layer for efficient training.
- We conduct comprehensive experiments and ablation studies to verify the effectiveness and efficiency of Diga.

2 RELATED WORK

2.1 Deep Learning in Anti-Money-Laundering

Supervised ML models have been thoroughly studied in AML literature, especially the most-used SVM, random forests, and boosting algorithms [25]. The major viewpoint indicates that the high false positive rate and the incapability of catching new money-laundering modes hinder their application in industry [6, 18]. Recently, researchers have paid more attention to the DL methods in AML. Paula et al.[27] incorporate AutoEncoder to model the transaction patterns. The anomaly transaction is inferred by measuring the distance between the latent embedding of the testing sample and other normal embeddings. Han et al. [9] invoke RNNs, convolutional neural networks, and natural language processing to assist experts in collecting money-laundering knowledge, which saves 30% labor cost empirically. With the development of GNN, researchers leverage the graph structure of transaction networks for a more distinguishable representation of accounts in the embedding space. Alarab et al. [1] firstly evaluate the competence of GCN with Multi-layer perception (MLP) for illicit Bitcoin transactions. Meanwhile, Weber *et al.* [26] proposed Evolve-GCN, which combines the strength of GNNs on topology and RNNs on temporal features and achieves SOTA performance in supervised learning.

Although SOTA methods in AML have made considerable progress in detection accuracy, they are still stuck in the research stage because the explainability and scalability required in the banking industry are out of scope [13]. Meanwhile, semi-supervised learning is a promising solution for both inheriting the labeled expert knowledge and watching new unseen modes simultaneously [3].

2.2 Graph Anomaly Detection with GNNs

GNNs have been proven as an effective embedding approach for detecting graph anomalies in many domains, such as biological, E-commerce, and social networks [28, 39, 44]. The message-passing mechanism can generate high-quality node embeddings by aggregating the node attributes based on the local topology [14]. We categorize the major solutions into two branches: **1)** The reconstruction-based detection first encodes the graph into a high-dimensional latent space. Then with the latent embeddings, the decoder reconstructs the original graph structure and node attributes. By comparing the difference in the reconstructed graph, the hidden anomalies with high reconstruction error can be discovered [15]. ComGA [22] encodes both node community-specific representation via deep MLP and node attribute via a tailored GNN and then reconstructs the graph-wise structure and node-wise attributes. AEGIS [5] invokes graph adversarial learning with generator and discriminator to distinguish generated anomalies against the normal nodes. However, large amounts of trainable parameters in multiple networks lead to a high computation cost and scalability problem of the current reconstruction-based method [38]. **2)** The clustering-based detection, represented by DeepFD [33] and semi-GCN [16], adopts an unsupervised learning manner. After the embedding by GNN, various clustering algorithms are applied in a hypersphere space to detect outliers or anomaly regions based on distance-based measurement. This unsupervised approach relies on a strong assumption on defining the boundary of outliers and still suffers a high false positive rate because of the absence of expert knowledge.

Most SOTA graph anomaly detection methods focus on the node or edge level [15]. There is still a huge gap between the existing subgraph detection techniques and the emerging demands for detecting anomalous group behaviors in the domains like AML [23]. The over-smoothing issue of GNN is also unsolved when directly applying it to a large-scale transaction graph [8].

3 PROBLEM DEFINITION

The transaction graph $\mathcal{G}$ consists of the set of accounts $\mathcal{U} = \{u_i \mid i = 1, 2, ...\}$ as the nodes and the set of transactions $\mathcal{R}$ as the edges. The node attribute vector $\boldsymbol{x}_i$ represents the features of each account u_i, such as the registration location, daily transaction frequency, monthly transaction volume, etc. The binary label y_i is defined by AML experts and represents whether the account u_i is money-laundering suspicious or not.

Definition 1. *(Semi-supervised Graph Anomaly Detection) Given the graph $\mathcal{G}$ with partially labeled nodes $\mathcal{U}_{labeled} \subset \mathcal{U}$, the main task is to compute the anomaly score of the unlabeled nodes and rank them in descending order.*

4 METHODOLOGY

In this section, we introduce the overall framework of Diga and three novel techniques. For a clear presentation of each module, we illustrate the general steps of AML inference with Diga in Figure 1.

Table 1: Major notations

Notation	Description
u_i	A node of user account
K	# total hop in subgraph sampling and GNN aggregation
N_c^k	The set of k-hop neighbors of center node $c \in \mathcal{U}$
$\mathcal{G}_c$	The subgraph surrounding center node c
$\boldsymbol{x}_i$	Original attribute vector of single node u_i
$\boldsymbol{X}_c$	Original node attribute matrix consists of $\{\boldsymbol{x}_i \mid \forall u_i \in \mathcal{G}_c\}$
$r(c, u_i)$	Residue value of node u_i w.r.t center node c
$\pi(c, u_i)$	Reserve value of node u_i w.r.t center node c
$\boldsymbol{Z}_c^t$	Noised node attribute matrix at diffusion timestep t
$\boldsymbol{h}_i^k$	Hidden embedding of node u_i on k_{th} GNN layer

4.1 Subgraph Sampler with Biased PageRank

The main objective of the subgraph sampler is to extract adaptive subgraph $\mathcal{G}_c$ surrounding a center user account c efficiently. Each subgraph contains the most representative neighbors and reserves the locality characteristic of the graph structure. Most existing subgraph-level GNNs reduce the subgraph sampling problem to a single-source Personalized PageRank (PPR) problem [21, 34]. However, the current PPR-based methods face significant obstacles in online AML systems: In the inference of anomaly accounts, current sampling methods operate on the transaction network with billions of nodes to monitor, the computation cost is unaffordable, e.g., FORA is the SOTA of single-source PPR [34], which takes about 1.6 seconds to generate one 2-hop subgraph of a center node and takes around 18.5 days to scan 1 million users, which much exceeds the timeliness requirement of detecting money laundering. Therefore, we solve the AML subgraph sampling with the biased K-hop PageRanks by allocating PPR value with forward pushes and reducing the number of random walks because the walks on large graphs

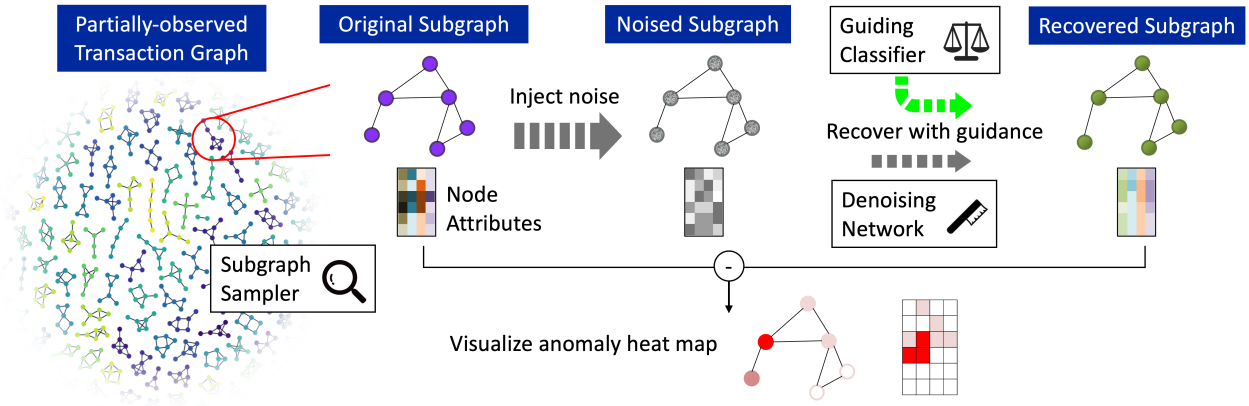

Figure 1: AML Inference with Diga: The subgraph sampler adaptively extracts a subgraph surrounding a center node with the biased K-hop PageRank. A series of Gaussian noises are injected into the original subgraph. The noised subgraph can be recovered in the reverse denoising process with the cooperation of the guiding classifier and the denoising network. Eventually, the anomaly score is measured by the reconstruction error between the original subgraph against the anomaly-free one. As an explanation, the corresponding anomaly heat map on the attribute level can be visualized.

are computational-expensive. Empirically, the average duration is shortened to 0.2 seconds per subgraph.

The subgraph sampler in Diga is extended from the one-hop PPR model Baton in [21] to a K-hop version. Diga performs forward pushes and random walks to allocate the residue value $r(c, u_i)$ and the reserve value $\pi(c, u_i)$ of each node u_i in the K-hop neighborhood N_c^K surrounding the center node c. Then, the PPR value $PPR(c, u_i)$ of each node w.r.t. to the center node can be estimated with these two indicators. The same dimensionality of the subgraph is conducive to an accurate graph recovery in the following diffusion model, so the maximal size of subgraphs is fixed with the hyperparameter M. The pseudocode of this sampling pipeline is shown in Algorithm 1. Starting from each center node $c \in \mathcal{U}$, we perform a one-step forward push to allocate a reserve value based on its residue. Specifically, the source node transfer the ρ portion of its residue to its one-hop neighbors and reserve the rest portion as its reserve. After K-hop propagation, each node in the K-hop subgraph gets its reserve share $\pi(c, u_i)$. Afterward, $r(c, u_i) \cdot L(c)$ random walks are executed from each node. The total number of random walks $L(c)$ is in direct proportion to the degree of the center node $d(c)$, referring to the formula in [21]. Based on the secondary distribution of residues with these random walks, the PRR importance w.r.t the center node can be estimated using the final reserve value. Then, the adaptive subgraph can be constructed with the top@M important nodes.

One significant distinction in the transaction network is that many accounts, such as merchants, retailers, and Uber drivers, have extremely large node degrees but less information related to money-laundering behaviors. Thus, instead of using a constant proportion ρ, a biased proportion function $\rho(u_i)$ for a biased PPR estimation by adjusting the allocation of residues in the forward push. The proportion is computed as:

$$\rho(u_i) = \sigma(\boldsymbol{a}_\rho \cdot \boldsymbol{x}_i + b_\rho), \quad with\ \boldsymbol{a}_\rho = [f_1(\cdot), f_2(\cdot), ...]^T \tag{1}$$

where σ is the Sigmoid activation function, the function vector $\boldsymbol{a}_\rho$ and the scalar b_ρ are manually defined to set various operations on each attribute. This revision on the allocation proportion opens a window for incorporating expert knowledge into node sampling by squeezing the sampling frequency of redundant information. For a fair evaluation, we only add a negative weight on the transaction frequency and prune out the same nodes with a large degree for all baselines in experiments.

Algorithm 1 K-hop PPR for subgraph sampling

Require: Transaction graph $\mathcal{G}$, Center node c, Hop K, Allocation proportion ρ, Sampling size M and # random walks $L(c)$

$\forall u_i \in N_c^K$, $\pi(c, u_i) = 0$ and $r(c, u_i) = 0$
$r(c, c) = 1$ # initial residue of the center node
while $\exists\, u_i \in N_c^K, r(c, u_i) > \frac{d(u_i)}{\rho(u_i) \cdot L(c)}$ and $d(u_i) > 0$ **do**
  for $\forall u_j \in N_{u_i}^1$ **do** # one-step forward push
    $r(c, u_j) = r(c, u_j) + (1 - \rho(u_i)) \frac{r(c, u_i)}{d(u_i)}$
  $\pi(c, u_i) = \pi(c, u_i) + \rho(u_i) \cdot r(c, u_i)$
  $r(c, u_i) = 0$
$\forall u_i \in N_c^K$, $PPR(c, u_i) = \pi(c, u_i)$
for $\forall u_i \in N_c^K$ and $r(c, u_i) > 0$ **do**
  for 1 to $r(c, u_i) \cdot L(c)$ **do**
    Execute a random walk from u_i on $\mathcal{G}$
    if the random walk terminates at u_j and $u_j \in N_c^K$ **then**
      $PPR(c, u_j) = PPR(c, u_j) + \frac{1}{L(c)}$
Sample $\mathcal{G}_c \subset \mathcal{G}$ with the top@M nodes ranking by $PPR(c, u_i)$
Concatenate $\boldsymbol{X}_c$ with set of attribute vectors $\{\boldsymbol{x}_i | \forall u_i \in \mathcal{G}_c\}$
return $\mathcal{G}_c$, $\boldsymbol{X}_c$

4.2 Guided Diffusion for Graph Recovery

The guided diffusion module is intended to operate graph recovery on each subgraph $\mathcal{G}_c$ and outputs the anomaly score of the center node and the corresponding heat map. Besides the totally different data structure between images and graphs, the technical challenge of the diffusion process on such an uncharted domain is how to enlarge the reconstruction error of anomaly subgraphs while maintaining the fidelity of those benign subgraphs. Therefore, we design two GNNs networks: The denoising network focuses

on reconstructing the subgraph as similar as possible to the original subgraph. The guiding classifier will change the recovering direction if the subgraph is predicted as an anomaly.

Following the general design of denoising diffusion models in computer vision [4, 30], the guided graph recovery in Diga also contains two inverse Markov chains: the forward diffusion process and the reverse denoising process. At each timestep t in the forward diffusion process, a Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ is injected into the attribute matrix $\boldsymbol{X_c}$ of the subgraph surrounding the center node c following a predefined variance schedule $\beta^1, ..., \beta^T$:

$$q(Z_c^t | Z_c^{t-1}) := \mathcal{N}(Z_c^t; \sqrt{1-\beta^t}\, Z_c^{t-1},\ \beta^t \boldsymbol{I}) \tag{2}$$

where $Z_c^0 = \boldsymbol{X_c}$. The hyperparameter T defines the total diffusion timestep from the original attribute matrix $\boldsymbol{X_c}$ to a totally noised matrix Z_c^T that follows a standard Gaussian distribution. After the recursion of (2), the noised attribute matrix Z_c^t is computed as:

$$Z_c^t = \sqrt{\bar{\alpha}^t} Z_c^0 + \sqrt{1-\bar{\alpha}^t}\, \boldsymbol{\epsilon},\ \ with\ q(Z_c^t | Z_c^0) := \prod_{t=1}^{t} q(Z_c^t | Z_c^{t-1}) \tag{3}$$

where $\alpha^t := 1 - \beta^t$ and $\bar{\alpha}^t := \prod_{s=1}^{t} \alpha^s$. In the denoising process, the denoising GNN network $\boldsymbol{\epsilon}_\theta(\mathcal{G}_c, Z_c^t, t)$ is trained to revert the diffusion process by sampling Z_c^{t-1} from a diagonal Gaussian distribution p_θ, given the sampled subgraph structure $\mathcal{G}_c$ and the partially-noised attribute matrix Z_c^t at timestep t, formulated as:

$$p_\theta(Z_c^{t-1} | Z_c^t) := \mathcal{N}(Z_c^{t-1};\ \boldsymbol{\mu}_\theta(\mathcal{G}_c, Z_c^t, t),\ \sigma_t^2 \boldsymbol{I}) \tag{4}$$

where σ_t^2 are set to the untrained constants β^t as the suggestion in [10], and the estimated mean value $\boldsymbol{\mu}_\theta$ can be calculated as:

$$\boldsymbol{\mu}_\theta(\mathcal{G}_c, Z_c^t, t) = \frac{1}{\sqrt{\alpha^t}}(Z_c^t - \frac{\beta^t}{\sqrt{1-\bar{\alpha}^t}} \boldsymbol{\epsilon}_\theta(\mathcal{G}_c, Z_c^t, t)) \tag{5}$$

With the predicted mean and variance of the Gaussian noise, the sampling of Z_c^{t-1} is computed by:

$$Z_c^{t-1} = \boldsymbol{\mu}_\theta(\mathcal{G}_c, Z_c^t, t) + \sigma_t \boldsymbol{z},\ \ with\, \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}) \tag{6}$$

With the recursion of this reverse process, the reconstructed node attribute matrix $\mathbf{Z}_c^0$ at step 0 can be obtained.

Next, we introduce the guiding classifier, which revises the predicted noise of denoising network $\boldsymbol{\epsilon}_\theta$ to recover the noised graph toward a determined "healthy" direction. Inspired by the conditioned image synthesis and the conditioned molecule generation [4, 11], the pretrained classifier can guide the sampling direction by adjusting the predicted noise towards a specific class [20]. For example, by applying a condition on the class *cartoon*, the diffusion model can transform a real photo into a cartoon-style painting [30].

In Diga, the guiding classifier $\boldsymbol{p}_\phi(y_c | \mathcal{G}_c, Z_c^t, t)$ also takes the subgraph structure and corresponding noised attribute matrix as input and predicts an anomaly label of the whole subgraph $\mathcal{G}_c$, which means it is a graph classifier with pooling layers introduced in the section 4.3. The supervising graph label y_c is aligned with the label of the center node c, where $y = 0$ represents a benign subgraph, and $y = 1$ is the label for suspicious subgraphs. As proven in the conditioned image generation in [4, 40], the output of denoising GNN is revised by the gradients $\nabla_{Z_c^t} log\, \boldsymbol{p}_\phi\,(y = 0 | \mathcal{G}_c, Z_c^t, t)$ to guide the denoising recovering process to generate the anomaly-free subgraph. And the gradient scale s controls whether the reverse process depends on the classifier decision or just keeps the graph similar to the original transaction graph.

After that, the anomaly score is measured by the $L2$ distance between the original attribute matrix and the recovered one. The anomaly heat map can be illustrated as an explanation, shown in Figure 4. We presented the pseudocode for guided diffusion in graph anomaly detection.

Algorithm 2 Graph anomaly detection with the guided diffusion

Require: Subgraph $\mathcal{G}_c$, Original attribute matrix $\boldsymbol{X_c}$, Total timestep T, Denoising network $\boldsymbol{\epsilon}_\theta$, Guiding classifier $\boldsymbol{p}_\phi$, Variance schedule $\{\beta^t\}$ and Gradient scale s

1: Forward diffusion $Z_c^T \leftarrow \sqrt{\bar{\alpha}^T} Z_c^0 + \sqrt{1-\bar{\alpha}^T}\, \boldsymbol{\epsilon}$
2: **For** denoising step t in $[T, T-1, ..., 1]$ **do**
3: $\quad \hat{\boldsymbol{\epsilon}} \leftarrow \boldsymbol{\epsilon}_\theta(\mathcal{G}_c, Z_c^t, t) - s\sqrt{1-\bar{\alpha}^t}\ \nabla_{Z_c^t} log\, \boldsymbol{p}_\phi\,(y = 0 | \mathcal{G}_c, Z_c^t, t)$
4: $\quad Z_c^{t-1} \leftarrow \frac{1}{\sqrt{\alpha^t}}(Z_c^t - \frac{\beta^t}{\sqrt{1-\bar{\alpha}^t}} \hat{\boldsymbol{\epsilon}}) + \sqrt{\beta^t}\boldsymbol{z},\ \ with\ \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$
5: Anomaly heat map $\boldsymbol{A_c} \leftarrow |\boldsymbol{X_c} - Z_c^0|$
6: Anomaly Score $g_c \leftarrow \|\boldsymbol{X_c} - Z_c^0\|_2^2$
7: **return** $\boldsymbol{A_c}$, g_c

4.3 Diga-GNN with Weight-sharing Mechanism

Two neural networks are involved in the guided diffusion model. Unlike the plentiful pretrained models in image classification, it is a heavy workload for training the guiding classifier additionally on the AML dataset. Meanwhile, the noise prediction and the subgraph classification task share a similar demand on the representation of graph topology and node aggregated features. Therefore, we design a novel Diga-GNN layer combining the graph pooling and node representation tasks into a unified network, which is essential for adapting the diffusion model on graph datasets. Diga-GNN with the corresponding alternative training pipeline can save the training time at most 53.5% compared to a separated training mode.

At each timestamp t, the input of Diga-GNN is the subgraph $\mathcal{G}_c$ with the partially noised node attribute matrix Z_c^t. Basically, Diga-GNN allows various GNN layers, such as GCN and GraphSage based on different dataset properties. Here, we use Graph Attention Network as an example for aggregating hidden node embedding of node u_i at layer k:

$$\boldsymbol{h}_i^k = \sigma(\boldsymbol{W}_1^{k-1} \boldsymbol{h}_i^{k-1} + \sum_{u_j \in \boldsymbol{N}_{u_i}^1} \gamma_{ij} \boldsymbol{W}_2^{k-1} \boldsymbol{h}_j^{k-1}) \tag{7}$$

where $\boldsymbol{N}_{u_i}^1$ is the node set of one-hop neighbors of node u_i, σ is activation function, and γ_{ij} is the attention weight between $\boldsymbol{h}_i^{k-1}$ and $\boldsymbol{h}_j^{k-1}$ referring to [31]. $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ is the trainable weight matrix. The timestep t is concatenated into the node features $\boldsymbol{h}_i^0 = CONCAT(Z_c^t, t)$. For the denoising network, after K-hop GNN aggregation, the latent embeddings are fed into MLP to predict the noise $\boldsymbol{\epsilon}_\theta$. As proven in [32], the denoising network can be trained by the Mean Square Error Loss between the ground truth of injected noises $\boldsymbol{\epsilon}$ and the network outputs. The loss function is formulated as follows:

$$\mathcal{L}_{DenoisingNetwork} = \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathcal{G}_c, Z_c^t, t)\|_2^2 \tag{8}$$

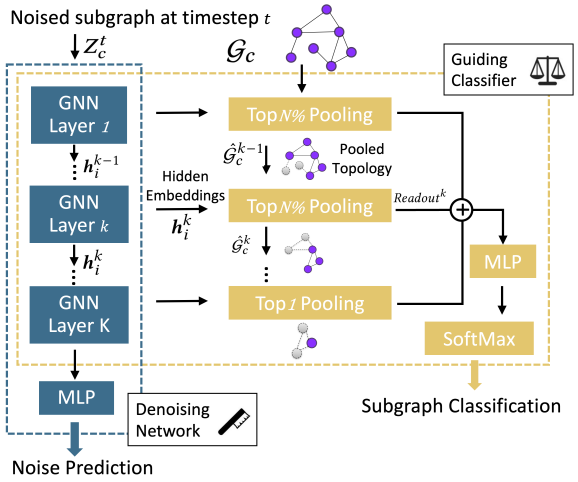

Figure 2: Diga-GNN with the weight-sharing mechanism

The graph pooling mechanism is essential to obtain reasonable graph representation [19]. The hierarchical pooling approach, which iteratively coarsens the graph into a new graph with a smaller size, is invoked here to generate the graph embedding for the binary classification task. In Diga-GNN, the weight-sharing mechanism is shown in Figure 2. We utilize the same interest in the topological structure in both networks. The GNN-aggregated hidden embeddings are shared on each layer. Based on the remained graph topology $\hat{\mathcal{G}}_c^{k-1}$ and the node embeddings from GNN layer $\boldsymbol{h}_i^k$, the pooling layer k calculates the node importance scores, and preserve the Top$N$% nodes. The remained graph topology $\hat{\mathcal{G}}_c^k$ waits for importance evaluation in the next layer with the new node embeddings from the next GNN layer. This iterative shrinking process will be terminated at the last layer, where only one node is preserved.

As the node importance calculation in the SOTA graph pooling methods [19, 41], the computation of importance score simultaneously considers the hidden embeddings and the raw attributes:

$$S_i^k = \lambda\, \sigma(\boldsymbol{h}_i^k) + (1-\lambda)\, \sigma(MLP(Z_c^t)),\ \ \forall u_i \in \hat{\mathcal{G}}_c^{k-1} \tag{9}$$

where $\hat{\mathcal{G}}_c^0 = \mathcal{G}_c$ and the constant λ is a hyperparameter. On each pooling layer, we apply both global max pooling and global mean pooling on remaining nodes to readout a subgraph-level embedding:

$$Readout^k = CONCAT(mean(\boldsymbol{h}_i^k), max(\boldsymbol{h}_i^k)),\ \ \forall u_i \in \hat{\mathcal{G}}_c^k \tag{10}$$

Ultimately, the final graph embedding is the sum of all intermediate readouts. With the individual MLP with SoftMax operation, the Log Likelihood is calculated for the binary classification. The guiding classifier can be optimized by the cross entropy loss compared to the batch graph label $\boldsymbol{y}_c$.

We design an alternative process for training the Diga-GNN. The hyperparameter τ decides the probability of taking the denoising network and corresponding objective function for the current data batch. Empirically, there is no significant relation between the probability τ with the model performance because the convergence direction of both branching networks is overlapping. The better graph representation by the shared GNN layers encourages both noise prediction and subgraph classification. However, as shown in Table 5, the convergence speed shows obvious preferences for more training batches on the denoising network with $\tau = 0.8$ based on the tuning result. The potential reason is that the prediction of the supervised classifier fluctuates with the unbalanced labels in the AML problem, making the training convergence unstable.

5 EXPERIMENTS

In this section, we state the experimental settings and compare Diga with SOTA methods. The ablation studies, parameter sensitivity analysis are also exhibited.

5.1 Experimental Settings

5.1.1 Dataset. We use the internal dataset from WeBank at three different scales. WeBank-large (WB-L) dataset is sampled from the real transaction data within three months. To investigate the model performance on various scales of datasets, we randomly select 12.5% and 50% volumes to build the WeBank-small (WB-S) and WeBank-medium (WB-M), respectively. Since the banking industry releases no other public dataset, we selected three publicly available datasets related to virtual currency data. OTC[1] and Alpha[2] dataset is a who-trusts-whom attributed datasets, which are widely used in the fraud user detection problem. We follow the setting in previous work to define the fraudulent users [36]. Elliptic[3] records the payment flows of Bitcoin. We choose the transaction period before time step 42 to prevent the impact of a complete change in network structure, which is caused by a sudden dark market closure at time step 43, as mentioned in [35]. The dataset statistics are shown in Table 3. For a fair comparison between node-level and subgraph-level methods, the nodes in each epoch consist of all labeled anomalies and the down-sampled normal accounts. Otherwise, the node-level baselines must iterate every single node over the graph. The final ratio of the normal class to the labeled anomaly class is about 9.5 : 1 as most graph anomaly detection models [22, 26, 28]. The datasets are divided into training, validation, and testing parts with the ratio of 80%, 10%, 10%.

5.1.2 Evaluation Metrics. AUC, the area under the ROC curve, combines the false positive rate and true positive rate in a visualizable way. It is applied the in most binary classification tasks as the AML detection [15]. Precision@n measures the proportion of true anomaly samples ranking in the top n with the highest anomaly score, widely used to evaluate the reconstruction-based inductive anomaly detection models [5, 43].

5.1.3 Baselines. We compare Diga with six SOTA baselines, including traditional supervised ML models, unsupervised DL models, and the SOTA baselines in GNN-based anomaly detection:

- **SVM** [7] is the baseline of the traditional supervised ML method. It shows priority in handling high-dimensional features of user accounts. In contrast, the scalability of SVM constrains its implementation in a real-world dataset.
- **XGBoost** [13] is a decision-tree-based boosting algorithm. As the supervised ML baseline, XGBoost has high classification accuracy and interpretability by feature importance analysis.
- **DeepFD** [33], as an unsupervised-learning and subgraph-level anomaly detection baseline, assumes that fraud detection problems like AML can be viewed as finding anomaly-dense blocks

[1] https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html
[2] https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html
[3] https://elliptic.co

Table 2: Overall graph anomaly detection performance comparison

Dataset	WeBank-small		WeBank-medium		WeBank-large		Elliptic		OTC		Alpha	
Metrices	Pre@100	AUC(%)	Pre@100	AUC(%)	Pre@100	AUC(%)	Pre@100	AUC(%)	Pre@100	AUC(%)	Pre@100	AUC(%)
SVM	0.162 ±0.021	54.0 ±1.5	-	-	-	-	-	-	0.519 ±0.002	68.3 ±0.1	0.477 ±0.001	63.8 ±0.3
XGBoost 2016	0.281 ±0.008	62.7 ±0.2	0.432 ±0.011	61.4 ±0.5	-	-	0.813 ±0.001	81.8 ±0.3	0.678 ±0.000	85.9 ±0.0	0.606 ±0.000	81.2 ±0.0
DeepFD 2018	0.126 ±0.037	49.2 ±2.1	0.666 ±0.038	69.4 ±3.6	0.613 ±0.049	65.7 ±4.6	0.688 ±0.033	72.3 ±3.9	0.656 ±0.031	74.6 ±2.2	0.369 ±0.022	54.7 ±3.4
SemiADC 2021	0.170 ±0.017	56.3 ±0.9	0.717 ±0.020	76.9 ±1.2	-	-	-	-	0.545 ±0.051	75.2 ±1.8	0.522 ±0.029	74.1 ±0.9
OCGTL 2022	0.267 ±0.006	60.3 ±0.1	0.802 ±0.009	77.7 ±0.1	0.751 ±0.012	73.4 ±0.4	0.912 ±0.007	90.3 ±1.1	0.622 ±0.001	83.6 ±0.6	0.683 ±0.005	80.7 ±0.1
ComGA 2022	0.331 ±0.020	71.9 ±0.6	0.804 ±0.016	80.8 ±2.0	-	-	0.855 ±0.029	85.2 ±3.6	0.701 ±0.005	87.0 ±2.4	0.634 ±0.016	84.1 ±1.9
Diga (Ours)	**0.383** ±0.004	**74.3** ±0.3	**0.900** ±0.010	**87.6** ±0.2	**0.949** ±0.008	**82.5** ±0.4	**0.981** ±0.005	**95.2** ±0.7	**0.730** ±0.010	**89.1** ±0.3	**0.724** ±0.019	**87.4** ±0.5

Table 3: Statistics of datasets

Dataset	WB-S	WB-M	WB-L	Elliptic	OTC	Alpha
# Nodes	58,409	233,638	467,277	203,769	5,858	3,754
# Edges	101,910	523,301	904,256	234,355	35,592	24,186
# Node Features	22	22	22	166	4	4
# Anomalies	478	1,830	3,676	4,545	178	102
Anomaly Ratio (%)	0.81	0.78	0.79	2.23	3.03	2.72
Avg. Degree	1.74	2.24	1.93	1.15	6.06	6.44

in the embedding space. Therefore, it embeds the graph via Autoencoder and uses DBSCAN to cluster the dense regions.

- **SemiADC** [24] focuses on semi-supervised anomaly detection with limited labelled data. SemiADC is the node-level baseline utilizing GAN to reconstruct the graph with the help of a graph generator and discriminator.
- **OCGTL** [28] is the baseline in graph-level anomaly detection, which combines graph transformation learning and one-class pooling to achieve the SOTA performance in both molecular and social network datasets.
- **ComGA** [22], as the SOTA GNN-based baseline, incorporates a deep graph convolution network to encoder the completed attributed graph and reconstruct the network by attribute decoder and structure decoder. Thus, it can measure the anomaly score with the reconstructed adjacent matrix and the feature matrix.

5.1.4 Architecture Details. The architectural details of Diga are listed here for easier reproduction. All hyperparameters are optimized with grid search. The hyperparameter sensitivity analysis on the hop number K, sampling size M, alternative learning rate τ, and total diffusion step T are shown in section 5.4. We keep consistent with the settings in most diffusion models, the variance schedule is a series of constants increasing linearly from $\beta^1 = 10^{-4}$ to $\beta^T = 0.02$. The gradient scale s is tuned in the range of $[0, 10]$ and its influence on model stability is also discussed in section 5.4. In Diga-GNN layer, the scoring weight λ is tuned to 0.6, and the preserving rate in pooling N is 50% as in [41]. The training will be terminated if no improvement is obtained in 50 epochs. This termination condition is also applied to all baselines. For the training process, the batch size is set to 64, the dropout is tuned to 0.1, and the learning rate is searched in $\{10^{-3}, 10^{-4}, 10^{-5}\}$. Online experiments are conducted in the production environment of WeBank, running on the NVIDIA A100 SXM4 and using Pytorch 1.4 as the computing infrastructure. The hyperparameters of baselines are aligned with original papers and optimized with the same validation set.

5.2 Overall Performance Comparison

The comparison of overall anomaly detection performance is shown in Table 2. All the results are averaged over five runs. Diga successfully outperforms six benchmarks by different margins. The observations from multiple aspects are concluded:

(1) On **larger** datasets, Diga shows more considerable improvement up to 26.4% margin in WeBank-large. The potential reason is that Diga pays more attention to the multipartite transaction modes inside a group of users based on **subgraph-level** detection. Notably, We feed the same subgraphs into the graph classifier OCGTL, which also achieves competent performance. In addition, the GNN-based method operating on the node level, such as ComGA and SemiADC, hardly converges in WeBank-Large. The over-smoothing issue of GNN tends to deteriorate on such a **sparse** and partially observed graph. As shown in the graph statistics, the average degree of transaction networks in WeBank and Elliptic is much smaller than other networks like the who-trust-who networks in OTC. The hidden feature is undistinguishable after multi-layers aggregation. The superiority of subgraph-level methods on large datasets verifies that handling money-laundering behaviors as a subgraph-level problem is promising instead of considering each account separately.

(2) Compared to the supervised classification models (SVM, XGBoost) or the pure unsupervised models (DeepFD), the **semi-supervised** methods show advantages in AML anomaly detection. Using graph reconstruction and weakly-supervised guidance to detect anomalies inductively is practicable. The main reason is that the unbalanced distribution between positive and negative labels limits the traditional supervised classifier. On the other hand, money laundering is a behavior with strong repetitiveness and regularity. The limited labels concentrate key business knowledge, enhancing the recognition of similar patterns. Notably, it is ineffective to find abnormal groups using clustering methods such as DeepFD. This implies that the money laundering group is not represented as a dense region in the embedding space, which is different from other graph anomaly problems.

(3) As for facing the **unbalanced** data, Diga proved that learning the normal data distribution through the diffusion model and learning expert knowledge with the supervised classifier effectively enlarge the reconstruction error between the original and recovered

subgraph. This can also be concluded from the more remarkable improvement in Precision@100. Compared to other generative models, GANs in ComGA and AutoEncoder-Decoder in SemiADC, the guided diffusion module can better rank labeled anomaly accounts over normal accounts.

(4) Diga does not outperform the baseline by a considerable margin on the OTC and Alpha datasets because there are few node attributes on the who-trust-who networks. The additional node attributes, such as node degree and average score, are not enough for the diffusion model to conduct effective graph recovery. A further study about Diga on other applications like fraud detection on social networks and emergent event on traffic networks is promising.

5.3 Ablation Study

In this section, we conduct the ablation study on anomaly detection performance and training efficiency to evaluate the functionality of each module proposed in Diga. We design three ablation models to evaluate their functionality in both effective and efficient aspects. We replace the biased K-Hop PPR with random sampling to build the subgraph in the model **w/o Sampling**. In the diffusion module, we remove the guiding classifier to build **w/o Guiding**, where the noises predicted by the denoising network are directly utilized in the graph reconstruction. This ablation mainly tests whether the reconstructed graph can find unseen anomaly patterns in a pure unsupervised environment. The last ablation, **w/o Sharing**, is intended to evaluate the difference between training with Diga-GNN or training two networks separately. Their performances are compared to the full Diga framework, as shown in Table 4.

Table 4: Performance comparison of ablation models

Dataset	WB-S	WB-M	WB-L	Avg. gain on AUC (%) of Diga compared to ablations
OCGTL	60.3	77.7	73.4	
ComGA	71.9	80.8	-	
w/o Sampling	73.4	82.1	78.0	+ 4.8 %
w/o Guiding	70.6	78.9	70.4	+ 11.2 %
w/o Sharing	**75.2**	**89.7**	**85.2**	- 2.3 %
Diga (Ours)	74.3	87.6	82.5	

In general, the average gain of Diga compared to each ablation verifies our design intentions. The biased K-hop PPR has a considerable improvement over random sampling. We also test the performance margin on the OTC dataset reaching 10.2%, which indicates that the effectiveness of adaptive sampling is related to the average degree of the network. Since the average degree of the transaction network is small, only a few choices can be selected in a K-hop neighborhood, making a similar output subgraph compared to the random sampling. Secondly, there is an apparent drop in AUC if the guiding classifier is removed. The large gain till 11.2% states the importance of incorporating expert knowledge for the observed modes in AML scenarios. However, the acceptable detection accuracy of **w/o Guiding** verifies that the anomaly parts can hardly be reconstructed so that new anomaly patterns can be discovered. This accords with the assumption in most unsupervised generative models [15].

As shown in Table 5, the sharing of the GNN layers in Diga is essential for the guided diffusion model in graph-based applications, which halves the entire training cost. This effect is more striking in the larger datasets at the cost of a few performance drops of about 2.3%. We also exhibit the relation between the training convergence and the probability τ of alternative training. For the detailed analysis, please refer to the last paragraph in section 4.3.

Table 5: Training efficiency comparison of ablation models and various alternative training rate τ

Dataset	WB-S	WB-M	WB-L	Avg. gain on training time (h) of Diga compared to ablations
OCGTL	2.10	3.8	4.2	
ComGA	0.36	3.2	>24	
w/o Sampling	0.30	0.76	1.02	- 15.9 %
w/o Guiding	**0.24**	**0.62**	**0.84**	- 41.2 %
w/o Sharing	0.50	1.50	3.16	+ 53.5 %
Diga ($\tau = 0.2$)	0.44	1.02	1.50	+ 18.9 %
Diga ($\tau = 0.5$)	0.43	0.95	1.46	+ 15.4 %
Diga ($\tau = 0.8$)	0.36	0.82	1.22	

5.4 Sensitivity Analysis

In this section, we demonstrate a particular discussion on the significant parameters of Diga. The hop number K and the sampling size M controls the aggregating range of each account on the whole network. As shown in Figure 3, there is a strong correlation between the AUC performance with the subgraph size. Generally, the 2-hop detecting horizon with a sampling size of 30 is optimal considering the trade-off between the performance and training cost. Interestingly, on the Elliptic dataset, the performance decreases if the sampling size exceeds the total number of existing nodes in the neighborhood. The Bitcoin transaction dataset barely forms a network structure but a long chain instead, because the average degree is only 1.15. That is also the reason for the depressed performance in ComGA, which calculate the anomaly score based on the reconstructed adjacency matrix. This design might not be suitable for the AML dataset with a sparse adjacent matrix. Thus, Diga utilizes the informative node attribute matrix as the material for reconstruction.

We also illustrate the sensitivity analysis of hyperparameters in the guided diffusion process, the total timestep T, and the gradient scale s. Firstly, the dataset with more node features, which means a larger node attribute matrix $\boldsymbol{X}_c$, prefers a larger diffusion step at the cost of longer training duration. Intuitively, a longer noise schedule is required to transform the larger matrix into a total Gaussian distribution completely. Secondly, the gradient scale s controls how much the guiding classifier affects the reserve denoising process. As the conclusion in the image synthesis problem, a larger gradient scale can promote a more precise generation toward the conditioned class [4, 37]. However, in graph anomaly detection, we found that the larger gradient scale also increases the variance of model performance in the significant test. The potential reason is that a sharp modification of denoising signals might impact the reconstruction of those unseen but regular subgraphs. Furthermore, the false positive rate might increase due to the intense intervention of the guiding classifier.

5.5 Inductive Detection Scenarios

In this section, we conduct studies on the model performance in the inductive setting with limited training samples. We randomly extract four training sets with different sizes from WeBank-medium

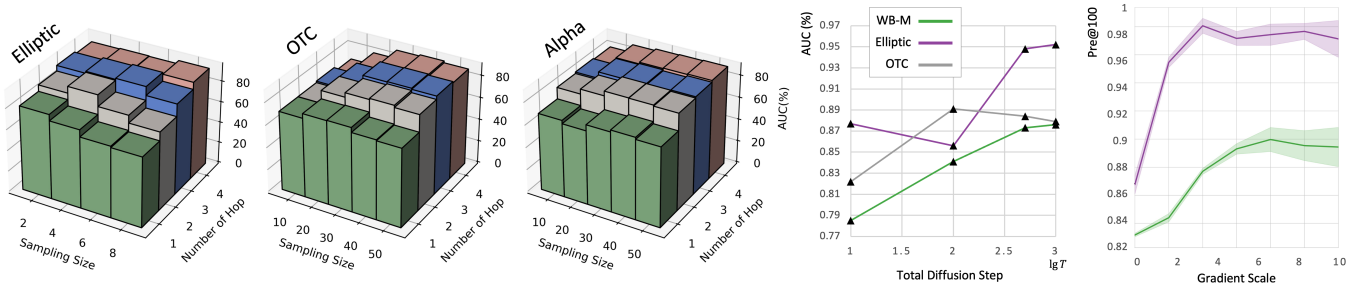

Figure 3: Parameter sensitivity analysis on the sampling range, the total diffusion step, and the gradient scale

and evaluate models with the same testing set. The results are shown in Table 6. Diga maintains a considerable performance with fewer training samples. Compared to the baselines, Diga achieved larger performance improvement up to 21.7% with the decline of the training set ratio. Supervised OCGTL requires ample labeled anomalies, while unsupervised ComGA requires more benign samples. Both of them suffer a sharp decline with an inadequate training set. The semi-supervised learning shows its robustness by considering both observed and unseen anomalies. This is more suitable for the AML problem, where the training samples take an extremely small proportion of the completed transaction networks, i.e., the trained model operates on a total inductive environment.

Table 6: Performance with various ratios of training set

Ratio of Training set	50%	60%	70%	80%
OCGTL	61.9	68.3	74.2	77.7
ComGA	63.5	72.3	81.2	80.8
Diga (Ours)	77.3	82.3	86.5	87.6
Improve.	21.7 %	13.8 %	6.5 %	8.4 %

5.6 Visualization of Anomaly Heat Map

We visualize one heat map in Figure 4, where each row represents a user account, and each column is one specific attribute. Because of the intellectual property of expert knowledge, we cannot provide a detailed description of each node feature, which is also not available in Elliptic. Intuitively, there are large reconstruction errors on the two labeled anomaly accounts, implying that guided diffusion can find specific reasons for each suspicious account on the attribute level, e.g., the darkest attribute in the black frame of Figure 4 might describe the midnight transaction frequency, which instructs the expert to investigate the midnight transaction records of this account and find concrete evidences. In addition, we leave the statistical evaluation of the explanation quality in future work, because heavy human-involved evaluation is infeasible with these confidential transaction data. We will generalize our model to broader scenarios like recommendations and analyze its performance from the view of explainable AI.

6 CONCLUSION AND FUTURE WORK

In this work, the proposed Diga model firstly sheds light on utilizing the probabilistic diffusion model in graph anomaly detection and

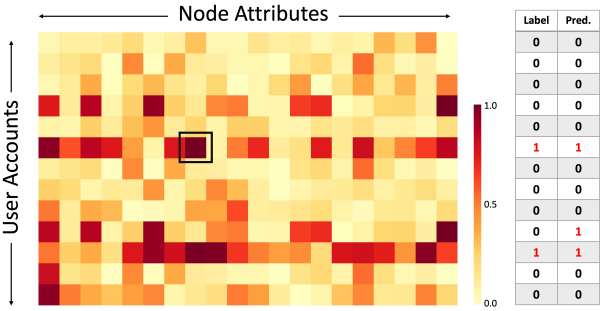

Figure 4: Visualization of anomaly heat map

addresses the major challenge of deploying DL models into the AML scenario in the banking industry. Several juicy takeaways are summarized: **1)** Subgraph-level recovery is practical on large, sparse, partially-observed transaction graphs, which requires the model to have stability in the inductive settings. **2)** A semi-supervised approach like the guided diffusion in Diga shows superiority in AML because the labeled suspicious account can boost model performance on observed money-laundering modes and simultaneously maintain the ability to catch new patterns in the inductive setting. **3)** Guided diffusion is feasible in conditioned graph generation, and the weight-sharing mechanism is significant. Otherwise, the additional training of the classifier is time-consuming for each specific graph-based task without public-available pretrained models. In future work, we will extend Diga into a general solution of graph anomaly detection, such as frauds in a social network and erupting events in a traffic network.

ACKNOWLEDGEMENT

Lei Chen's work is partially supported by the National Science Foundation of China (NSFC) under Grant No. (U22B2060, 61729201), the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C2004-21GF, AOE Project AoE/E-603/18, RIF Project R6020-19, Theme-based Project TRS T41-603/20R, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF Grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant, HKUST-WeBank Joint Research Lab Grant and HKUST Global Strategic Partnership Fund (2021 SJTU-HKUST).

REFERENCES

[1] I. Alarab, S. Prakoonwit, and M. I. Nacer. 2020. Competence of Graph Convolutional Networks for Anti-Money Laundering in Bitcoin Blockchain. In *ICMLT 2020: 2020 5th International Conference on Machine Learning Technologies*.

[2] Tianyi Chen and Charalampos Tsourakakis. 2022. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2762–2770.

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

[4] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 8780–8794.

[5] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2021. Inductive anomaly detection on attributed networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 1288–1294.

[6] O. Garcia-Bedoya, O. Granados, and J. C. Burgos. 2020. AI against money laundering networks: the Colombian case. *Journal of Money Laundering Control* ahead-of-print, ahead-of-print (2020).

[7] Jorge Guevara, Olmer Garcia-Bedoya, and Oscar Granados. 2020. Machine learning methodologies against money laundering in non-banking correspondents. In *International Conference on Applied Informatics*. Springer, 72–88.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems* 2017-December, Nips (2017), 1025–1035.

[9] J. Han, U. Barman, J. Hayes, J. Du, E. Burgin, and D. Wan. 2018. NextGen AML: Distributed Deep Learning based Language Technologies to Augment Anti Money Laundering Investigation. In *Meeting of the Association for Computational Linguistics*.

[10] J. Ho, A. Jain, and P. Abbeel. 2020. Denoising Diffusion Probabilistic Models. (2020).

[11] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. 2022. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*. PMLR, 8867–8887.

[12] Stockholm International Peace Research Institute. 2022. SIPRI Yearbook 2022. (2022).

[13] Martin Jullum, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, and Johannes Lorentzen. 2020. Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control* 23, 1 (2020), 173–186.

[14] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. 2022. Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges. (2022).

[15] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. 2022. Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges. *IEEE Access* (2022).

[16] A. Kumagai, T. Iwata, and Y. Fujiwara. 2021. Semi-supervised Anomaly Detection on Attributed Graphs. In *International Joint Conference on Neural Network*.

[17] Dattatray Vishnu Kute, Biswajeet Pradhan, Nagesh Shukla, and Abdullah Alamri. 2021. Deep learning and explainable artificial intelligence techniques applied for detecting money laundering–a critical review. *IEEE Access* 9 (2021), 82300–82317.

[18] Nevine Makram Labib, Mohammed Abo Rizka, and Amr Ehab Muhammed Shokry. 2020. Survey of machine learning approaches of anti-money laundering techniques to counter terrorism finance. In *Internet of Things—Applications and Future*. Springer, 73–87.

[19] C. Liu, Y. Zhan, C. Li, B. Du, J. Wu, W. Hu, T. Liu, and D. Tao. 2022. Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. (2022).

[20] M. Lucic, M. Tschannen, M. Ritter, X. Zhai, and S. Gelly. 2019. High-Fidelity Image Generation With Fewer Labels. (2019).

[21] Siqiang Luo, Xiaokui Xiao, Wenqing Lin, and Ben Kao. 2019. BATON: Batch One-Hop Personalized PageRanks with Efficiency and Accuracy. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[22] Xuexiong Luo, Jia Wu, Amin Beheshti, Jian Yang, Xiankun Zhang, Yuan Wang, and Shan Xue. 2022. ComGA: Community-Aware Attributed Graph Anomaly Detection. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 657–665.

[23] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z. Sheng, Hui Xiong, and Leman Akoglu. 2021. A Comprehensive Survey on Graph Anomaly Detection with Deep Learning. *IEEE transactions on knowledge and data engineering* (2021).

[24] Xuying Meng, Suhang Wang, Zhimin Liang, Di Yao, Jihua Zhou, and Yujun Zhang. 2021. Semi-supervised anomaly detection in dynamic communication networks. *Information Sciences* 571 (2021), 527–542.

[25] Marvin Oeben, Jeroen Goudsmit, and Elena Marchiori. 2019. Prerequisites and AI challenges for model-based Anti-Money Laundering. In *IJCAI 2019 Workshop*.

[26] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5363–5370.

[27] Ebberth L. Paula, Marcelo Ladeira, Rommel N. Carvalho, and Thiago Marzagao. 2017. Deep Learning Anomaly Detection as Support Fraud Investigation in Brazilian Exports and Anti-Money Laundering. In *IEEE International Conference on Machine Learning and Applications*.

[28] Chen Qiu, Marius Kloft, Stephan Mandt, and Maja Rudolph. 2022. Raising the Bar in Graph-level Anomaly Detection. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. 2196–2203.

[29] Omri Raiter. 2021. Applying Supervised Machine Learning Algorithms for Fraud Detection in Anti-Money Laundering. *Journal of Modern Issues in Business Research* (2021), 14–26.

[30] J. Song, C. Meng, and S. Ermon. 2020. Denoising Diffusion Implicit Models. (2020).

[31] Petar Velikovi, G. Cucurull, A. Casanova, A. Romero, P Liò, and Y. Bengio. 2017. Graph Attention Networks. (2017).

[32] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2022. DiGress: Discrete Denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734* (2022).

[33] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, and Jilong Wang. 2018. Deep Structure Learning for Fraud Detection. In *2018 IEEE International Conference on Data Mining (ICDM)*.

[34] S. Wang, R. Yang, X. Xiao, Z. Wei, and Y. Yin. 2017. FORA: Simple and Effective Approximate Single-Source Personalized PageRank. In *Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*.

[35] M. Weber, G. Domeniconi, J. Chen, Dki Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. 2019. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *KDD Workshop on Anomaly Detection in Finance, Anchorage, AK, USA* (2019).

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

[37] Julia Wolleb, Florentin Bieder, Robin Sandkühler, and Philippe C. Cattin. 2022. Diffusion Models for Medical Anomaly Detection. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022*. 35–45.

[38] Julian Wyatt, Adam Leach, Sebastian M Schmon, and Chris G Willcocks. 2022. AnoDDPM: Anomaly Detection With Denoising Diffusion Probabilistic Models Using Simplex Noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 650–656.

[39] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. 2022. GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation. In *International Conference on Learning Representations*.

[40] S. Yang, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, and B. Poole. 2020. Score-Based Generative Modeling through Stochastic Differential Equations. (2020).

[41] Liang Zhang, Xudong Wang, Hongsheng Li, Guangming Zhu, Peiyi Shen, Ping Li, Xiaoyuan Lu, Syed Afaq Ali Shah, and Mohammed Bennamoun. 2020. Structure-feature based graph self-adaptive pooling. In *Proceedings of The Web Conference 2020*. 3098–3104.

[42] Shengming Zhang, Yanchi Liu, Xuchao Zhang, Wei Cheng, Haifeng Chen, and Hui Xiong. 2022. CAT: Beyond Efficient Transformer for Content-Aware Anomaly Detection in Event Sequences. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4541–4550.

[43] Yu Zheng, Ming Jin, Yixin Liu, Lianhua Chi, Khoa T. Phan, and Yi-Ping Phoebe Chen. 2021. Generative and Contrastive Self-Supervised Learning for Graph Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. https://doi.org/10.1109/TKDE.2021.3119326

[44] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1, January (2020), 57–81.