GAD-NR: Graph Anomaly Detection via Neighborhood Reconstruction

Amit Roy*
Purdue University
roy206@purdue.edu

Juan Shu
Purdue University
shu30@purdue.edu

Jia Li
Hong Kong University of Science and Technology
jialee@ust.hk

Carl Yang
Emory University
j.carlyang@emory.edu

Olivier Elshocht
Sony R&D Center Brussels Laboratory
Olivier.Elshocht@sony.com

Jeroen Smeets
Sony R&D Center Brussels Laboratory
jeroen.smeets@sony.com

Pan Li
Georgia Institute of Technology
panli@gatech.edu

ABSTRACT

Graph Anomaly Detection (GAD) is a technique used to identify abnormal nodes within graphs, finding applications in network security, fraud detection, social media spam detection, and various other domains. A common method for GAD is Graph Auto-Encoders (GAEs), which encode graph data into node representations and identify anomalies by assessing the reconstruction quality of the graphs based on these representations. However, existing GAE models are primarily optimized for direct link reconstruction, resulting in nodes connected in the graph being clustered in the latent space. As a result, they excel at detecting cluster-type structural anomalies but struggle with more complex structural anomalies that do not conform to clusters. To address this limitation, we propose a novel solution called GAD-NR, a new variant of GAE that incorporates neighborhood reconstruction for graph anomaly detection. GAD-NR aims to reconstruct the entire neighborhood of a node, encompassing the local structure, self-attributes, and neighbor attributes, based on the corresponding node representation. By comparing the neighborhood reconstruction loss between anomalous nodes and normal nodes, GAD-NR can effectively detect any anomalies. Extensive experimentation conducted on six real-world datasets validates the effectiveness of GAD-NR, showcasing significant improvements (by up to 30%↑ in AUC) over state-of-the-art competitors. The source code for GAD-NR is openly available. Importantly, the comparative analysis reveals that the existing methods perform well only in detecting one or two types of anomalies out of the three types studied. In contrast, GAD-NR excels at detecting all three types of anomalies across the datasets, demonstrating its comprehensive anomaly detection capabilities.

* Corresponding Author : Amit Roy.

WSDM '24, May 4th–8th, 2024, Mérida, Yucatán, Mexico

ACM ISBN 978-1-4503-XXXX-X/18/06.
https://doi.org/10.1145/3616855.3635767

CCS CONCEPTS

• **Computing methodologies** → *Anomaly Detection; Unsupervised Learning; Neural networks..*

KEYWORDS

Anomaly Detection, Graph Neural Network, Auto-Encoder

ACM Reference Format:
Amit Roy*, Juan Shu, Jia Li, Carl Yang, Olivier Elshocht, Jeroen Smeets, and Pan Li. 2024. GAD-NR: Graph Anomaly Detection via Neighborhood Reconstruction. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24), March 4–8, 2024, Merida, Mexico.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3616855.3635767

1 INTRODUCTION

Anomaly Detection aims to identify entities that deviate significantly from the norm, which has been used for a variety of applications, such as revealing fraudulent or spam activity in social networks [26, 59, 68, 72, 79, 80] and financial transactions networks [10, 18, 19, 53, 62, 65, 70].

Unlike anomaly detection methods for tabular and time-series data, Graph Anomaly Detection (GAD) [3, 48, 64] poses additional challenges. Graph data is often multi-modal, containing information from both node/edge attributes and topological structures. This complexity makes it difficult to find a unified definition of anomalies for graph-structured data and to design a principled algorithm for detecting them.

Due to the inherent multi-modality of graph-structured data, anomalies on graphs can be grouped into three categories: contextual, structural, and joint-type, as illustrated in Fig. 1. Contextual anomalies refer to nodes whose attributes are vastly different from those of regular nodes, such as spammers or fake account holders in social media networks [29, 32, 75]. Structural anomalies refer to nodes with different connectivity patterns compared to other nodes, such as a group of malicious sellers exchanging fake reviews with super dense connections [69] or bots retweeting the same tweet forming a densely connected co-retweet network [21, 28].

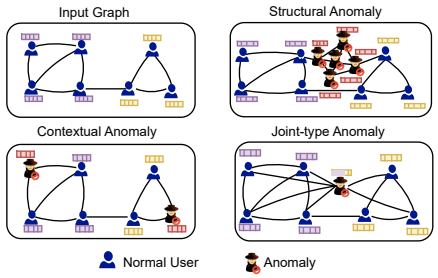

Figure 1: Contextual anomalies are feature-wise different, structural anomalies form dense subgraphs in the network and joint-type anomalies connect with many nodes with different features.

Joint-type anomalies are those that can only be identified by considering both attributes and connectivity patterns, such as a node that is sending a large number of phishing emails to users across different communities in an email network [35, 52]. To identify all these types of anomalies, we need a powerful model to capture attribute information, connectivity patterns, and most importantly the correlation between them.

However, current GAD approaches [3, 44, 48] only perform well to detect one or two types of these anomalies but not all of them. Some GAD approaches only leverage network structure, which cannot detect contextual anomalies. Examples include the methods to check centrality measures or clustering coefficients [49, 60], based on factorization of the adjacency matrix [67], and performing network clustering [77]. Some approaches check the distribution of node features to detect anomalies [5, 42], such as using the k-nearest neighbor algorithm on node features, to detect nodes that are isolated from others. These approaches fail to detect anomalies other than contextual anomalies.

Recently, autoencoders have been widely employed for anomaly detection [7, 15, 20, 36, 58]. The rationale is that autoencoders leverage neural networks to reduce the dimension of the data. Anomalies are often sparse in the data and hence such a data compression process tends to record only the principal part of the data and automatically exclude sparse anomalies. Therefore, one can use the obtained compressed data representations to approximately reconstruct the normal data but not the anomalies. Monitoring the reconstruction loss can thus identify those anomalies from the normal data. For GAD, Graph Auto-Encoders (GAEs) have been proposed to leverage Graph Neural Networks (GNNs) [25, 38, 76] to encode both graph structure and node attributes, which have recently been used to detect anomalies on graphs [15, 20, 36].

However, current GAE-based methods [15, 20, 36] often adopt a strategy of reconstructing direct links between nodes based on their representations, which brings the nodes close to each other in the latent space that are originally connected in the graph structure. Such a proximity-driven loss to reconstruct graph structures may be effective to detect structural anomalies that are inherently clustered together in the graph. However, they fail to detect joint-type anomalies that are not naturally clustered. Intuitively, joint-type anomalies rely on the entire neighborhoods for correct detection, because the information of which nodes are connected and the attributes on these neighboring nodes is useful for the detection.

In this paper, we address the current limitation and propose a novel framework **G**raph **A**nomaly **D**etection via **N**eighborhood

Approach	Contextual Anomaly	Structural Anomaly	Joint-type Anomaly
Structure-based SCAN [77] and others [49, 60, 67]	×	✓	×
Feature-based LOF [5], IF [42], MLPAE [58]	✓	×	×
GAE with proximity driven loss AnomalyDAE [20] , GCNAE [36] DOMINANT [15]	✓	✓	×
GAD-NR (ours)	✓	✓	✓

Table 1: SOTA methods perform well either on contextual or densely connected structural anomalies whereas GAD-NR with its entire neighborhood reconstruction principle finds advantages for detecting both types of anomalies along with joint-type anomalies which are the nodes that connect a large number of nodes with different features.

Reconstruction (**GAD-NR**). GAD-NR extends a recently-proposed neighborhood reconstruction-based GAE model, namely NWR-GAE [66] to address fundamental problems in GAD. Specifically, rather than using a proximity-driven loss to recover direct links, GAD-NR imposes the dimension-reduced node representations to reconstruct the entire neighborhoods, i.e., the receptive fields that are encoded/compressed by GNNs into the node representations. Specifically, GAD-NR aims to reconstruct the information of one's own attributes, its connectivity pattern, and the attributes of its neighboring nodes. By checking different types of reconstruction losses, GAD-NR can detect all three types of anomalies.

The key novelty of GAD-NR is that it is the first work that identifies neighborhood reconstruction as a powerful metric for GAD, which fundamentally differs from previous GAE models that adopt the metric of link reconstruction/prediction. Moreover, GAD-NR also advances technical aspects of the backbone model NWR-GAE [66] directly for GAD tasks, which yields substantial improvements in stability, scalability, and accuracy. Specifically, GAD-NR adopts Gaussian approximation of neighbors' features distributions, which not only substantially reduces the computation cost of NWR-GAE but also avoids learning a too expressive model that risks overfitting the anomalous behaviors in the data. This non-trivial change improves NWR-GAE originally proposed for the unique purpose of dimension reduction now suitable for GAD tasks.

We extensively compare GAD-NR with state-of-the-art (SOTA) models on six real-world graph anomaly detection datasets that have been benchmarked recently [44]. GAD-NR outperforms all baselines significantly (by up to 30%↑ in AUC) over five among these six datasets by following the settings in [44]. We also evaluate and demonstrate the capability of GAD-NR on detecting each of the three types of anomalies.

Note that in real-world applications, the types of anomalies are often unknown. The significance of GAD-NR is that it allows detecting the real-world anomalies across different datasets (in [44]) with one fixed hyperparameter configuration, which illustrates the robustness of GAD-NR. Further ablation studies also justify the effectiveness and computational efficiency of Gaussian approximation adopted by GAD-NR for GAD when being compared with NWR-GAE [66].

The contributions of this paper can be summarized as follows:

- We designed a novel framework GAD-NR for graph anomaly detection. GAD-NR leverages the reconstruction loss of the entire

neighborhood of a node from the node representation, which in principle can detect all three types of anomalies in Fig. 1.
- Technically, GAD-NR adopts a Gaussian approximation of the distribution of neighbors' representations and computes a closed-form KL divergence as the reconstruction loss, which substantially improves the scalability and effectiveness of the approach.
- Extensive experiments on six real-world networks demonstrate the effectiveness of GAD-NR compared to SOTA baselines, and the rationale of the design specifics of GAD-NR.

2 RELATED WORKS

We put previous methods for GAD into three categories as follows.

Structure-only-based methods: Traditional graph anomaly detection focuses on detecting only structural anomalies. Many works in this category leverage spectral analysis of the adjacency matrix and its variants [31, 50]. Recent methods define structural similarity measures for anomalies and then perform clustering approaches for detection [54, 77]. Statistical features computed based on the graph structure such as in/out degrees, total weights of edges, number of neighbors of a node, or dense subgraphs can be utilized for GAD [2, 17, 28]. However, these structure-based methods are only able to detect structural anomalies. They may detect some joint-type anomalies but they tend to make a slot of false alarms as they miss the information from node attributes.

Traditional methods for GAD over attributed networks: In real-world applications, most of the graphs have node attributes (features). Nodes with inconsistent attributes have a high chance to be an anomaly node. Moreover, considering the information on node attributes along with structure helps to locate anomalies more accurately. Detecting anomalies in attributed networks can be achieved by clustering methods [9, 56], interaction with human experts [16], group merging techniques [83]. Network embedding methods [23, 57, 63] can also be applied to GAD on attributed graphs [6, 8]. Network embeddings can be paired with anomaly detection techniques for tabular data such as density-based approaches [5], and distance-based techniques [1, 42] to find node anomalies on graphs. However, these approaches, since they process graph structure separately with node attributes, often fail to capture the synergy of graph structure and node attributes and may be suboptimal for GAD in some cases.

Deep learning based GAD approaches: Auto-Encoder framework that focuses on extracting principal components from the data via deep learning has been extensively applied in anomaly detection [7, 15, 20, 36, 47]. Applying traditional autoencoders to node attributes [58] can only detect contextual anomalies. GAE built upon GNNs can combine node attributes and graph structure properly and can detect anomalies based on checking the reconstruction loss of node attributes or links [15, 20, 36]. But these works do not reconstruct the entire neighborhood for GAD. Rather, they use reconstruction error, and estimating Gaussian mixture density is also applied for GAD [41]. Some works view nodes with multiple views and a node may or may not be considered an anomaly in different views. These nodes hold attributes from multiple views of the identity. To capture such multi-view information, multiple GNNs are often applied [46, 55, 61, 73, 74] for anomaly detection. GNNs have also been applied to detect anomalies in multiple scales [24], and to detect anomalies and solve recommendation tasks simultaneouly [71, 81]. More involved techniques such as self-supervised learning [13, 30, 33, 45, 78, 82] and reinforcement learning [16, 39, 51] have also been recently applied to GAD.

3 NOTATIONS AND PROBLEM FORMULATION

In this work, we focus on detecting anomalous nodes over attributed static graphs. An attributed graph $G = (V, E, X) \in \mathcal{G}$ consists of a vertex set $V = \{1, 2, \cdots, N\}$ and an edge set E. $X = [\cdots x_u^\top \cdots]^\top \in \mathbb{R}^{|V| \times k}$ collect all node attributes and $x_u \in \mathbb{R}^k$ is the attribute for node u. The degree of node u is denoted as d_u. This work focuses on unsupervised anomaly detection. Each node u has an anomaly label y_u where $y_u = 0$ or $y_u = 1$ implies node u is normal or anomalous respectively. The goal is to design a detection method $f(G) : \mathcal{G} \rightarrow \{0, 1\}^N$ that associates each nodes with a label. However, these node labels are assumed to be unknown when designing f.

Let $\mathcal{N}_u$ be the set of 1-hop neighbor nodes of node u. Let $\tilde{\mathcal{N}}_u$ be an augmented set of 1-hop neighborhood of node u that includes the attribute of node u, the set of the attributes of its neighbors, i.e., $\tilde{\mathcal{N}}_u \triangleq (x_u, \{x_v | v \in \mathcal{N}_u\})$. Our assumption to detect anomalous nodes is that given the label y_u, the distribution $\mathbb{P}(\tilde{\mathcal{N}}_u | y_u)$ are different across norms and anomalies. Here, we consider just one-hop neighborhood as a proof of concept, which is also often adequate for use cases in practice [4]. The neighborhoods considered can be extended to the multi-hop case, while extra computation costs need to be paid in that scenario.

4 METHODOLOGY

In this section, we first provide the motivation of our method by narrating the potential drawbacks of previous graph auto-encoder methods. Then, we introduce GAD-NR which is based on neighborhood reconstruction.

4.1 Motivations

AutoEncoder (AE) is an easy-to-use and effective framework for anomaly detection. The motivation of AE is to perform dimension reduction by compressing the high dimensional input data into a low dimensional latent representation [27] via an encoder and reconstructing the original input with the help of a decoder. AE can be used for anomaly detection because such dimension reduction is expected to capture the principal properties of the data mostly corresponding to the normal data points. The data points that cannot be properly reconstructed via the decoder, i.e., with larger reconstruction losses tend to be anomalies.

Graph AutoEncoder (GAE) is used to perform dimension reduction of graph data via a Graph Neural Network (GNN) as the encoder [37]. Given a graph $G = (V, E)$, GAE encodes graph data into node representations $\{h_v | v \in V\}$. The decoder of current GAE methods is to reconstruct the graph structure and node attributes from these node representations. Regarding graph-structure reconstruction, it typically relies on a mapping from the representations of two nodes to 0 or 1 that indicates whether there is an edge between them [15, 20], e.g., comparing $h_u^\top h_v$ with some threshold θ to reconstruct the edge. However, this procedure can only preserve proximity information of nodes in the graph, i.e., pushing node representations close if the corresponding nodes are directly

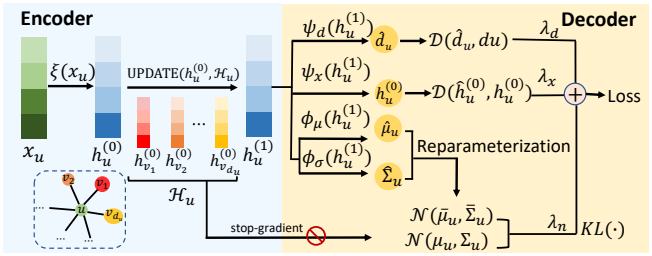

Figure 2: Model architecture of GAD-NR. The encoder (left) performs dimension reduction with an MLP followed by a message passing GNN to obtain the hidden representation of a node. The decoder (right) reconstructs the self features and the node degree via MLPs and estimates the neighbor feature distribution with an MLP-predicted re-parameterized Gaussian distribution. Reconstructions of the self features and the node degree are optimized with MSE-loss whereas the KL-divergence between the ground truth and the learned neighbors' feature distribution is used for the optimization of the distribution estimation.

connected in the graph, which may miss useful information for detecting anomalies. Moreover, by checking the reconstruction loss, one may only tell whether an edge is an anomaly. To detect node anomalies that are often more useful in practice, one needs to aggregate the reconstruction losses of edges into the node level, and how to properly aggregate these losses is not a trivial problem by itself and often depends on heuristics.

4.2 GAE via Neighborhood Reconstruction

Our strategy to overcome the drawback of traditional GAEs lies in the first-principle idea of autoencoders. Autoencoders aim to perform dimension reduction of the data with the least loss to recover the original data. GAE encodes each node's attributes and the attributes of the nodes in its one-or-several-hop neighborhood into a node representation. Therefore, the node representation should be able to reconstruct the neighborhood and its attributes with the least loss. This idea leads to the design of GAE in this work. The model architecture is illustrated by Fig. 2 and describes the pseudocode in Algorithm 1.

4.2.1 The encoder. The encoder $\Phi(\cdot)$ follows the common pipeline of message passing GNN [22] e.g. GCN [38] or GraphSAGE [25]. A GNN will further iteratively aggregate the representations from the neighbors and combine them with one's own representation to update the representation. Specifically, let $h_u^{(0)} = x_u$. For $l = 0, 1, ..., L-1$,

$$h_u^{(l+1)} = \text{UPDATE}\left(h_u^{(l)}, \text{AGG}\left(\{h_v^{(l)} : v \in \mathcal{N}_u\}\right)\right), \tag{1}$$

The AGG function aggregates messages from the neighbors and the UPDATE function updates the node representations. Note that in practice, if the node attribute x_u is extremely high dimensional and sparse, a random linear projection is used to encode them into a dense low-dimensional representation $h_u^{(0)} = \xi(x_u)$.

4.2.2 The decoder. Our decoder is designed based on the first principle of designing an autoencoder [27]. We are supposed to reverse the procedure of Eq. (1) by using $h_u^{(L)}$ to reconstruct all the information within the L-hop neighborhood of u. In practice, it is computationally heavy for large L. In this work, we focus on reconstructing the information within just the one-hop neighborhood as a proof of concept, which we find is practically sufficient for GAD.

Algorithm 1 GAD-NR : Graph Anomaly Detection via Neighborhood Reconstruction

Input: Graph $G(V, E)$, Input Feature X, Anomaly Label Y
Encoder:
for $u \in V$ **do**
  $h_u^{(0)} = \xi(x_u)$
  $h_u^{(1)} = \text{UPDATE}\left(h_u^{(0)}, \text{AGG}\left(\{h_v^{(0)} : v \in \mathcal{N}_u\}\right)\right)$
end for
Decoder:
for $u \in V$ **do**
  $\hat{h}_u^{(0)} = \psi_x(h_u^{(1)}), \hat{d}_u = \psi_d(h_u^{(1)})$
  $\mu_u = \text{stop-gradient}(\frac{1}{d_u}\sum_{v\in\mathcal{N}_u} h_v^{(0)})$
  $\Sigma_u = \text{stop-gradient}(\frac{1}{d_u-1}\sum_{v\in\mathcal{N}_u}(h_v^{(0)} - \mu_u)(h_v^{(0)} - \mu_u)^\top)$
  $\hat{\mu}_u = \phi_\mu(h_u^{(1)}), \hat{\Sigma}_u = \text{diag}(\exp(\phi_\sigma(h_u^{(1)})))$
  for $i = 1$ to q **do** ▷ Reparameterization
    $\bar{h}_i = \text{FNN}(z_i), z_i \sim \mathcal{N}(\hat{\mu}_u, \hat{\Sigma}_u)$
  end for
  $\bar{\mu}_u = \frac{1}{q}\sum_{i=1}^{q}\bar{h}_i, \bar{\Sigma}_u = \frac{1}{d_u-1}\sum_{i=1}^{q}(\bar{h}_i - \bar{\mu}_u)(\bar{h}_i - \bar{\mu}_u)^\top$
  Weighted Loss Function
  $\mathcal{L} = \lambda_x \sum_{u\in V}\mathcal{D}\left(h_u^{(0)}, \hat{h}_u^{(0)}\right) + \lambda_d \sum_{u\in V}\mathcal{D}\left(d_u, \hat{d}_u\right) + \lambda_n \text{KL}(\mathcal{N}(\mu_u, \Sigma_u)||\mathcal{N}(\bar{\mu}_u, \bar{\Sigma}_u))$
end for

The information within just the one-hop neighborhood $\bar{\mathcal{N}}_u$ consists of the attributes of the center node $h_u^{(0)}$ and the set of attributes of the direct neighbors of the center $\mathcal{H}_u = \{h_v^{(0)} | v \in \mathcal{N}_u\}$.

Self reconstruction: In order to reconstruct the attributes of the center node, we design a simple decoder that takes $h_u^{(L)}$ as input and reconstructs $h_u^{(0)}$ by a multi-layer perception (MLP) $\hat{h}_u^{(0)} = \psi_x(h_u^{(1)})$. Then, the self-reconstruction loss for node u can be calculated as:

$$\mathcal{L}_u^x = \mathcal{D}\left(h_u^{(0)}, \hat{h}_u^{(0)}\right) \tag{2}$$

where $\mathcal{D}(\cdot,\cdot)$ is a distance function such as L2-distance that measures the discrepancy between the original attributes and the reconstructed attributes.

Neighborhood reconstruction: It is far from trivial to decode the set $\mathcal{H}_u$ from the compressed $h_u^{(L)}$. The difficulties come from two aspects. Firstly, the size of the set might vary across node $u \in V$. Secondly, the elements in the set do not have an order. Using an MLP to decode a set of a variable size from $h_u^{(L)}$ is impossible.

Our idea is inspired by the recent work NWR-GAE [66]. We regard the set $\mathcal{H}_u = \{h_v^{(0)} | v \in \mathcal{N}_u\}$ as d_u many I.I.D. samples from a distribution $\mathbb{P}_u$. In fact, the empirical distribution of the elements in $\mathcal{H}_u$ is $\mathbb{P}_u^{\text{emp}}(h) = \frac{1}{d_u}\sum_{v\in\mathcal{N}_u}\delta(h - h_v^{(0)})$ where $\delta(\cdot)$ is the dirac delta function. In this sense, we can decompose the neighborhood information into two parts, namely the number of neighbors (i.e. the node degree) d_u and the distribution of neighbor's representations $\mathbb{P}_u$. The reconstruction procedure should reconstruct these two parts of information properly.

Node degree reconstruction. To reconstruct node degree d_u, we use another MLP that follows $\hat{d}_u = \psi_d(h_u^{(L)})$.

Then the node degree reconstruction loss for node u is:

$$\mathcal{L}_u^d = \mathcal{D}\left(d_u, \hat{d}_u\right) \tag{3}$$

Here, we just use ℓ_2-loss as the metric $\mathcal{D}$, though as node degrees are non-negative integers, we can also adopt discrete distributions such as Poisson distribution to model it.

Neighbors' representation distribution reconstruction. To reconstruct the distribution $\mathbb{P}_u$ from the node representation $h_u^{(L)}$, we first map $h_u^{(L)}$ to an estimation of the distribution $\hat{\mathbb{P}}_u$. As we do not know $\mathbb{P}_u$ in the population level, the first direct idea is to reconstruct the empirical distribution $\mathbb{P}_u^{\text{emp}}$ by following NWR-GAE [66]. The Wasserstein distance between $\hat{\mathbb{P}}_u$ and $\mathbb{P}_u^{\text{emp}}$ is adopted as the reconstruction loss in NWR-GAE [66]. However, the computation of such a loss has a huge overhead, because it needs to solve a matching problem based on the Hungarian algorithm [34], which is of complexity $O(d_u^3)$. Moreover, empirically, we observe that reconstructing such an empirical distribution is likely to overfit the anomalies, which actually does harm to anomaly detection tasks.

Therefore, we propose to reconstruct a multi-variate Gaussian approximation of $\mathbb{P}_u$. Specifically, given $\mathcal{H}_u = \{h_v^{(0)} | v \in \mathcal{N}_u\}$, we estimate the mean and covariance matrix of the neighbors' representations by following:

$$\mu_u = \frac{1}{d_u} \sum_{v \in \mathcal{N}_u} h_v^{(0)}, \; \Sigma_u = \frac{1}{d_u - 1} \sum_{v \in \mathcal{N}_u} (h_v^{(0)} - \hat{\mu}_u)(h_v^{(0)} - \hat{\mu}_u)^\top \tag{4}$$

Then, we map $h_u^{(L)}$ to a multi-variate Gaussian distribution $\hat{\mathbb{P}}_u$ through the following procedure. We sample q neighborhood features $\bar{h}_1, \cdots, \bar{h}_q$ by transforming samples $z_1, \cdots, z_q$ from the distribution $\mathcal{N}(\hat{\mu}_u, \hat{\Sigma}_u)$ via a fully-connected neural network (FNN). Here the parameters $\hat{\mu}_u, \hat{\Sigma}_u$ are determined by

$$\hat{\mu}_u = \phi_\mu(h_u^{(L)}), \quad \hat{\Sigma}_u = \text{diag}(\exp(\phi_\sigma(h_u^{(L)}))), \tag{5}$$

where $\phi_\mu(\cdot)$ is an MLP, and each entry of $\phi_\sigma(h_u^{(L)})$ is non-negative, which includes an MLP followed by $\exp(\cdot)$. Then, we estimate the mean and the covariance matrix of reconstructed neighbors' features based on $\bar{\mu}_u = \frac{1}{q}\sum_{i=1}^q \bar{h}_i$ and $\bar{\Sigma}_u = \frac{1}{d_u - 1}\sum_{i=1}^q (\bar{h}_i - \bar{\mu}_u)(\bar{h}_i - \bar{\mu}_u)^\top$, respectively.

Given the two groups of parameters (μ_u, Σ_u) and $(\bar{\mu}_u, \bar{\Sigma}_u)$ for multi-variate Gaussian distributions, we adopt the KL divergence between these two distributions to measure the reconstruction loss:

$$\begin{aligned} &\mathcal{L}_u^n = \text{KL}(\mathcal{N}(\mu_u, \Sigma_u) || \mathcal{N}(\bar{\mu}_u, \bar{\Sigma}_u)) = \\ &\frac{1}{2}[\log \frac{|\Sigma_u|}{|\bar{\Sigma}_u|} - p + \text{tr}(\bar{\Sigma}_u^{-1}\Sigma_u) + (\mu_u - \bar{\mu}_u)^\top \bar{\Sigma}_u^{-1}(\mu_u - \bar{\mu}_u).] \end{aligned} \tag{6}$$

where p is the dimension of representation. Note that μ_u and Σ_u should not allow the gradient to pass through as they provide supervision signals. In practice, we may encounter the case when d_u is smaller than p, which makes Σ_u in Eq. (4) not full-ranked and causes a numerical problem. Therefore, we add an identity matrix to the covariance matrices, $\Sigma_u \leftarrow \Sigma_u + cI$, $\bar{\Sigma}_u \leftarrow \bar{\Sigma}_u + cI$ for some constant $c > 0$ to compute Eq. (6).

Note that the complexity of the above computation including Eqs. (4),(5) and (6) is $O(d_u)$, which significantly reduces the complexity of the pipeline in [66].

The remaining challenge is that since node degrees vary across different nodes, the computation of Eq. (4) is irregular. For this, we extend the package adopted in principal neighborhood aggregation [14] to implement Eq. (4) efficiently in parallel across different nodes.

4.2.3 The overall reconstruction loss. The overall reconstruction loss is a combination of the losses to reconstruct node self attributes in Eq. (2), node degrees in Eq. (3), and neighbors' representation distributions in Eq. (6):

$$\mathcal{L} = \sum_{u \in V} \mathcal{L}'_u, \text{ where } \mathcal{L}'_u \triangleq \lambda_x \mathcal{L}_u^x + \lambda_d \mathcal{L}_u^d + \lambda_n \mathcal{L}_u^n, \tag{7}$$

and where λ_x, λ_d, and λ_n are the hyper-parameters that control the weights of different types of reconstruction losses.

4.3 Anomaly Detection

We may adopt $\mathcal{L}_u$ in Eq. (7) as the score to characterize how anomalous each node u is to be. The greater score means the encoded information is harder to be reconstructed, and thus the corresponding node is more likely to be an anomaly. We may also adopt different hyperparameters λ'_x, λ'_d, and λ'_n if we have different confidence or some prior knowledge about the type of anomaly to be detected. For example, if we tend to detect contextual anomalies, we can increase λ'_x. To encode such flexibility, we define the anomaly score $\hat{y}_u$ in the following general form

$$\hat{y}_u = \mathcal{L}'_u(\lambda'_x, \lambda'_d, \lambda'_n) = \lambda'_x \mathcal{L}_u^x + \lambda'_d \mathcal{L}_u^d + \lambda'_n \mathcal{L}_u^n, \tag{8}$$

ranking which tells the nodes that are more likely to be anomalies. Although different weights here emphasize the detection of different types of anomalies, in Sec. 5, we show that GAD-NR is robust to the selection of these weights, where a fixed choice of these weights is sufficient to outperform baselines to detect real-world anomalies across different datasets.

4.4 Improvements over NWR-GAE

Here, we would like to provide a more direct explanation how GAD-NR advances the idea of neighborhood reconstruction (previously proposed in NWR-GAE [65] for the purpose of dimension reduction instead of GAD) to better fit GAD tasks. NWR-GAE is built upon an optimal transport loss and needs to run a complicated Hungarian matching algorithm [34] for each node to reconstruct its neighbors' attributes to compute the loss function. Such complexity is $O(d^3)$ for a node of degree d. GAD-NR regards the representations of neighbors' attributes as samples from a Gaussian distribution and adopts KL divergence [12] between Gaussian distributions as the reconstruction loss, which has a closed form and has complexity $O(d)$. This approximation is crucial for GAD tasks:

NWR-GAE did not adopt such approximation because NWR-GAE aims to perform dimension reduction. Achieving a low reduction error is the ultimate goal of NWR-GAE. Therefore, NWR-GAE should be sufficiently expressive to make low-dimensional representations recover high-dimensional data. However, GAD tasks have different goals. A model for GAD should not be too expressive, and

otherwise risks overfitting the anomalies. GAD-NR just adopts the correct trade-off, where Gaussian approximation (by just checking the first and second moments of the distributions) is adopted, which not only improves anomaly detection accuracy but also substantially reduces computational complexity. Moreover, NWR-GAE also supports reconstructing multi-hop neighbors. However, we found that multi-hop reconstruction did not get obvious improvement in the task of GAD while introducing much computational overhead, so GAD-NR only considers the first hop in practice.

5 EXPERIMENT

In this section, we extensively compare GAD-NR with several baseline methods for graph anomaly detection. Specifically, we aim to answer the following questions:

- How does neighborhood reconstruction facilitate in the performance improvement of GAD-NR for GAD?
- Which part of the GAD-NR is important for different types of anomaly detection?
- How do important hyperparameters such as the size of hidden representations, and the weights before different types of reconstruction losses affect the performance of GAD-NR?
- How does the adopted Gaussian approximations of neighborhood feature distributions improve the running time efficiency of GAD-NR?

5.1 Datasets and Baselines

We incorporate six real-world datasets (Cora, Weibo, Reddit, Disney, Books, and Enron) and fourteen baseline anomaly detection models for our comparison following the BOND [43] paper. Among the baseline models, we included feature-based models LOF [5], IF [42], MLPAE [58] and structure-based AD models, SCAN [77]. Also, we performed comparisons of GAD-NR with models that focus on both structures and attributes via residual reconstruction error Radar [40] and ANOMALOUS [54]. Lastly, we incorporated some popular generative models for GAD, which include autoencoder architecture GCNAE [36], DOMINANT [15], DONE and AdONE [7], AnomalyDAE [20], adversarial learning-based method GAAN [11] and also contrastive learning-based methods CONAD [78].

5.2 Experimental Settings

Our first experimental setting follows the benchmark paper BOND [44]. Note that among the datasets, Weibo, Reddit, Disney, Books, and Enron have real-world anomaly labels. For the Cora dataset, there are no real benchmark anomaly labels, so we follow the benchmark paper BOND where the union of contextual and structural anomalies are considered anomaly labels for evaluation in the Cora dataset. The results are reported in Table 2. We call this setting the benchmark anomaly detection. Moreover, we attempt to study the performance to detect each type of anomalies separately, so for each dataset including those with real-world labels, we also inject contextual, structural, and joint-type anomalies for evaluation, which give the later results in Table 4. Due to the page limitation, we present contextual and merged the structural and joint-type anomaly detection results together in this work.

Contextual anomalies are nodes whose attributes are significantly different from their neighboring nodes. Hence, to generate this type of anomaly for a target node u, its feature x_u is replaced with another randomly sampled node v's feature x_v that has the largest Euclidean distance with x_u. Let n denote the number of contextual anomaly nodes and q denote the number of candidate nodes randomly sampled in the above procedure. Structural anomalies are nodes that are densely connected in contrast to sparsely connected normal nodes. To inject structural outliers, we consider m nodes at random and then make them fully connected and this process will be repeated for n times to generate n such cliques of size m. Following the BOND paper, we approximately set q and m as twice the avg. degree for most datasets. To add joint-type anomalies in different datasets, we choose n nodes randomly as anomalies. Then, we connect each of these n nodes with randomly sampled m other nodes. Therefore, those anomalous nodes can be treated as nodes with high degrees and connected to neighbors with different types of features. We utilized the PyGOD library [44] for the injection of contextual and structural anomalies and for running the baseline anomaly detection models.

Hyperparameter Tuning: In practice, we often do not know the anomaly labels to tune the parameters. Typically, the way to choose hyperparameters is based on some expert experiences and a good model should be robust by using such a set of hyperparameters. Hence, we fix GAD-NR's encoder as GCN with hidden dimension 16 (expect for Cora, where 128 is used) and fix the hyperparameters of the decoder as $\lambda_x = 0.8$, $\lambda_d = 0.5$, and $\lambda_n = 0.001$ to run the experiments for all datasets five times and report the averaged performance with standard deviation. We compare GAD-NR's performance obtained via this fixed hyperparameter setting with the averaged performance of baselines proposed in the benchmark work [43]. Our experiments demonstrate that our model GAD-NR can outperform the baselines by setting a set of hyperparameters that are not sensitive to the datasets, which makes the most practical sense.

To compare with the best performance of baselines reported in [43], we also performed a grid search of the hyperparameters of GAD-NR for each dataset as follows: 1) Self-attribute reconstruction weight, $\lambda_x \in \{0.1, 0.5, 0.8, 0.9\}$ and $\lambda'_x \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$, 2) Degree reconstruction weight, $\lambda_d \in \{0.1, 0.5, 0.8\}$ and $\lambda'_d \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$, 3) Neighborhood reconstruction weight $\lambda_n \in \{0.001, 0.5, 0.8\}$ and $\lambda'_n \in \{0.25, 0.5, 0.75,1.0, 1.25, 1.5, 1.75, 2.0\}$, 4) The number of dimension of the hidden layer, $d \in \{8, 16, 32, 64, 128\}$, 5) Encoder GNN, $\Phi \in \{$GCN, GraphSAGE, GIN$\}$.

Hardware: All the experiments are performed on a Linux server with a 2.99GHz AMD EPYC 7313 16-Core Processor and 1 NVIDIA A10 GPU with 24GB memory.

5.3 Evaluation Metric

We adopt the area under the ROC curve as the evaluation metric. The ROC curve is created by plotting the true positive rate against the false positive rate at various threshold settings. In the experiment, we regard the anomaly nodes as positive classes and compute AUC for it. AUC equals 1 means that the model makes a perfect prediction, and AUC equals 0.5 means that the model has no distinguishing ability. AUC is better than accuracy when evaluating the anomaly detection task since it is not sensitive to the imbalanced class distribution of the data.

Algorithm	Cora	Weibo	Reddit	Disney	Books	Enron
LOF	69.9 ± 0.0 (69.9)	56.5 ± 0.0 (56.5)	57.2 ± 0.0 (57.2)	47.9 ± 0.0 (47.9)	36.5 ± 0.0 (36.5)	46.4 ± 0.0 (46.4)
IF	64.4 ± 1.5 (67.4)	53.5 ± 2.8 (57.5)	45.2 ± 1.7 (47.5)	57.6 ± 2.9 (63.1)	43.0 ± 1.8 (47.5)	40.1 ± 1.4 (43.1)
MLPAE	70.9 ± 0.0 (70.9)	82.1 ± 3.6 (86.1)	50.6 ± 0.0 (50.6)	49.2 ± 5.7 (64.1)	42.5 ± 5.6 (52.6)	73.1 ± 0.0 (73.1)
SCAN	62.8 ± 4.5 (72.6)	63.7 ± 5.6 (70.8)	49.9 ± 0.3 (50.0)	50.5 ± 4.0 (56.1)	49.8 ± 1.7 (52.4)	52.8 ± 3.4 (58.1)
Radar	65.0 ± 1.3 (66.0)	**98.9 ± 0.1 (99.0)**	54.9 ± 1.2 (56.9)	51.8 ± 0.0 (51.8)	52.8 ± 0.0 (52.8)	80.8 ± 0.0 (80.8)
ANOMALOUS	55.0 ± 10.3 (68.0)	**98.9 ± 0.1 (99.0)**	54.9 ± 5.6 (60.4)	51.8 ± 0.0 (51.8)	52.8 ± 0.0 (52.8)	80.8 ± 0.0 (80.8)
GCNAE	70.9 ± 0.0 (70.9)	90.8 ± 1.2 (92.5)	50.6 ± 0.0 (50.6)	42.2 ± 7.9 (52.7)	50.0 ± 4.5 (57.9)	66.6 ± 7.8 (80.1)
DOMINANT	82.7 ± 5.6 (84.3)	85.0 ± 14.6 (92.5)	56.0 ± 0.2 (56.4)	47.1 ± 4.5 (54.9)	50.1 ± 5.0 (58.1)	73.1 ± 8.9 (85.0)
DONE	82.4 ± 5.6 (87.9)	85.3 ± 4.1 (88.7)	53.9 ± 2.9 (59.7)	41.7 ± 6.2 (50.6)	43.2 ± 4.0 (52.6)	46.7 ± 6.1 (67.1)
AdONE	81.5 ± 4.5 (87.4)	84.6 ± 2.2 (87.6)	50.4 ± 4.5 (58.1)	48.8 ± 5.1 (59.2)	53.6 ± 2.0 (56.1)	44.5 ± 2.9 (53.6)
AnomalyDAE	83.4 ± 2.3 (85.3)	91.5 ± 1.2 (92.8)	55.7 ± 0.4 (56.3)	48.8 ± 2.2 (55.4)	62.2 ± 8.1 (73.2)	54.3 ± 11.2 (69.1)
GAAN	74.2 ± 0.9 (76.1)	92.5 ± 0.0 (92.5)	55.4 ± 0.4 (56.0)	48.0 ± 0.0 (48.0)	54.9 ± 5.0 (61.9)	73.1 ± 0.0 (73.1)
GUIDE	74.7 ± 1.3 (77.5)	OOM_C	OOM_C	38.8 ± 8.9 (52.5)	48.4 ± 4.6(63.5)	OOM_C
CONAD	78.8 ± 9.6 (84.3)	85.4 ± 14.3 (92.7)	56.1 ± 0.1 (56.4)	48.0 ± 3.5 (53.1)	52.2 ± 6.9 (62.9)	71.9 ± 4.9 (84.9)
GAD-NR (w/o feat. recon.)	83.41 ± 2.18 (85.41)	69.64 ± 0.75 (70.44)	50.00 ± 0.44 (50.88)	74.11 ± 0.18 (76.17)	62.32 ± 3.41 (65.43)	70.20 ± 1.16 (71.72)
GAD-NR (w/o degree recon.)	82.25 ± 1.37 (83.04)	70.09 ± 1.20 (71.50)	49.01 ± 0.29 (50.05)	76.25 ± 0.37 (79.09)	64.08 ± 3.13 (68.94)	72.44 ± 1.33 (75.81)
GAD-NR (w/o neighbor recon.)	76.47 ± 3.57 (80.47)	69.10 ± 1.10 (70.25)	48.67 ± 2.04 (50.67)	60.69 ± 1.24 (63.69)	49.46 ± 2.09 (52.46)	56.01 ± 3.00 (59.01)
GAD-NR	**87.55 ± 2.56 (88.40)**	87.71 ± 5.39 (92.09)	**57.99 ± 1.67 (59.90)**	**76.76 ± 2.75 (80.03)**	**65.71 ± 4.98 (69.79)**	**80.87 ± 2.95 (82.92)**

Table 2: Performance comparison (ROC-AUC) of GAD-NR in benchmark anomaly detection for six different real-world datasets (injected anomaly for Cora dataset). For the results of baseline methods, we followed the BOND [43] paper where the *avg performance ± the STD of perf. (max perf.)* is reported. For our model GAD-NR, we fix hyperparameters $\lambda_x = 0.8$, $\lambda_d = 0.5$ and $\lambda_n = 0.001$ and report the *avg performance ± the STD of perf.* for all datasets including the best performance in each dataset with tuned hyperparameters. The best and second best performances are mentioned in bold and underlined respectively and *OOM_C* indicates out of memory with regard to GPU.

Comparison Type	Algorithm	Cora	Weibo	Reddit	Disney	Books	Enron
Performance Comparison	NWR-GAE	84.28 ± 0.06	73.68 ± 3.13	51.20 ± 1.16	75.56 ± 1.65	79.75 ± 2.48	80.24 ± 2.43
	GAD-NR	**87.55 ± 2.56**	**87.71 ± 5.39**	**57.99 ± 1.67**	**76.76 ± 2.75**	**65.71 ± 4.98**	**80.87 ± 2.9**
Running time Seconds per epoch	NWR-GAE	51.270	48.312	55.379	0.603	7.288	65.152
	GAD-NR	**2.35**	**0.544**	**0.095**	**0.035**	**0.0874**	**0.0109**

Table 3: Direct Performance Comparison between NWR-GAE [66] and our model GAD-NR

Algorithm	Cora	Weibo	Reddit	Disney	Books	Enron
MLPAE	88.90 ± 0.00	90.61 ± 0.02	51.91 ± 4.55	86.36 ± 0.00	53.00 ± 13.99	68.74 ± 16.08
SCAN	49.80 ± 0.50	48.46 ± 0.00	48.59 ± 0.00	62.81 ± 0.00	50.15 ± 0.00	40.41 ± 0.00
Radar	50.20 ± 0.60	72.31 ± 0.00	49.98 ± 0.00	79.89 ± 0.00	66.30 ± 0.00	79.51 ± 0.00
ANOMALOUS	51.10 ± 1.30	72.31 ± 0.00	49.65 ± 1.26	79.89 ± 0.00	66.30 ± 0.00	48.92 ± 0.85
GCNAE	88.90 ± 0.00	**90.79 ± 0.35**	52.02 ± 0.36	87.52 ± 2.32	40.77 ± 1.72	59.52 ± 15.23
DOMINANT	71.90 ± 6.60	57.07 ± 0.34	47.96 ± 0.39	65.62 ± 9.53	50.13 ± 5.33	63.84 ± 0.27
DONE	70.2 ± 8.30	81.75 ± 1.00	46.52 ± 1.47	69.26 ± 5.33	38.52 ± 2.08	59.90 ± 6.31
AdONE	73.90 ± 5.00	83.68 ± 0.47	46.61 ± 3.42	86.12 ± 1.83	65.00 ± 2.63	62.19 ± 5.37
AnomalyDAE	80.20 ± 2.80	79.40 ± 3.67	49.33 ± 2.28	81.02 ± 8.13	40.43 ± 18.56	68.12 ± 14.93
GAAN	88.70 ±0.10	90.68 ± 0.14	49.64 ± 1.12	86.39 ± 0.13	32.99 ± 22.38	44.78 ± 14.20
GUIDE	88.30 ± 0.80	OOM_C	OOM_C	78.40 ± 0.62	57.59 ± 0.08	OOM_C
CONAD	72.50 ± 5.80	56.62 ± 0.37	47.70 ± 0.08	59.67 ± 7.00	54.04 ± 7.22	71.23 ± 0.54
GAD-NR (w/o feat. recon.)	58.52 ± 2.32	57.66 ± 3.22	49.60 ± 2.11	82.98 ± 2.60	62.11 ± 5.01	68.27 ± 2.09
GAD-NR (w/o degree recon.)	73.04 ± 2.60	64.28 ± 6.35	54.10 ± 1.96	**91.10 ± 3.56**	60.07 ± 1.17	75.25 ± 3.57
GAD-NR (w/o neighbor recon.)	71.52 ± 3.59	61.45 ± 4.42	53.45 ± 1.15	57.11 ± 2.50	55.36 ± 3.19	68.23 ± 3.34
GAD-NR	**89.10 ± 3.10**	87.53 ± 3.54	**55.15 ± 2.41**	85.72 ± 1.31	**74.73 ± 1.50**	**85.79 ± 2.65**

Algorithm	Cora	Weibo	Reddit	Disney	Books	Enron
MLPAE	51.28 ± 0.43	50.42 ± 0.00	50.10 ± 0.52	58.77 ± 0.00	52.52 ± 2.48	48.30 ± 1.21
SCAN	82.35 ± 0.00	49.86 ± 0.00	**98.11 ± 0.00**	64.82 ± 0.00	62.21 ± 0.00	48.08 ± 0.00
Radar	62.37 ± 0.00	60.04 ± 0.00	65.70 ± 0.00	63.77 ± 0.00	32.69 ± 0.00	53.29 ± 0.00
ANOMALOUS	45.39 ± 1.15	60.04 ± 0.00	58.08 ± 29.77	63.77 ± 0.00	32.69 ± 0.00	52.81 ± 0.23
GCNAE	51.19 ± 0.00	50.66 ± 0.06	51.30 ± 0.43	52.82 ± 0.90	31.93 ± 0.35	47.33 ± 6.84
DOMINANT	77.59 ± 0.03	49.01 ± 0.92	93.18 ± 0.00	34.84 ± 4.38	63.99 ± 0.11	62.70 ± 0.10
DONE	72.34 ± 14.02	55.56 ± 0.90	65.61 ± 10.50	71.12 ± 1.40	87.86 ± 0.42	50.88 ± 2.51
AdONE	81.32 ± 1.39	59.23 ± 0.53	80.07 ± 2.26	71.43 ± 2.76	**88.30 ± 0.61**	55.93 ± 2.05
AnomalyDAE	80.73 ± 0.93	49.16 ± 1.33	47.03 ± 2.66	49.93 ± 6.26	32.25 ± 9.01	49.83 ± 1.89
GAAN	53.26 ± 0.84	53.09 ± 0.18	62.78 ± 4.85	60.20 ± 0.35	78.98 ± 1.00	59.88 ± 2.35
GUIDE	52.03 ± 2.36	OOM_C	OOM_C	55.20 ± 1.50	63.72 ± 0.89	OOM_C
CONAD	78.85 ± 0.02	55.48 ± 0.36	93.25 ± 0.17	34.49 ± 2.32	63.97 ± 0.31	59.98 ± 0.45
GAD-NR (w/o feat loss)	73.23 ± 0.61	50.81 ± 0.28	91.95 ± 0.03	73.23 ± 1.61	80.06 ± 5.49	79.18 ± 2.01
GAD-NR (w/o degree loss)	74.28 ± 1.67	53.33 ± 2.31	87.69 ± 6.47	66.47 ± 0.51	82.95 ± 0.30	72.19 ± 0.06
GAD-NR (w/o neigh loss)	67.51 ± 0.04	50.29 ± 1.11	91.34 ± 0.11	60.54 ± 2.06	56.28 ± 3.03	75.65 ± 1.39
GAD-NR	**83.55 ± 3.03**	**62.35 ± 1.05**	92.01 ± 0.73	**74.81 ± 4.39**	85.01 ± 7.90	**82.22 ± 2.14**

Table 4: Performance comparison (ROC-AUC) of GAD-NR in contextual (left) and structural + joint-type (right) anomaly detection for different real-world datasets. The best and second best performances are mentioned in bold and underlined respectively and *OOM_C* indicates out of memory with regard to GPU.

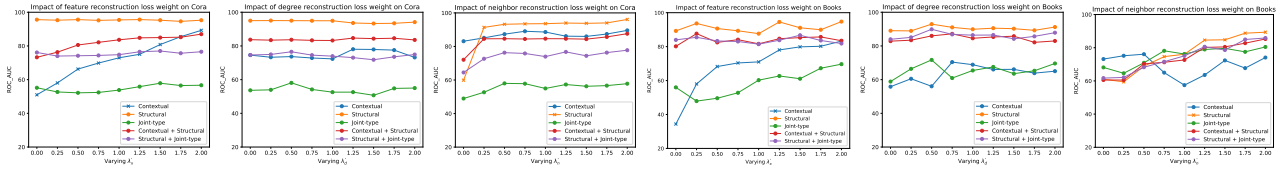

Figure 3: Impacts of varying feature reconstruction weight loss λ'_x, degree reconstruction weight loss λ'_d and neighbor reconstruction weight loss λ'_n in Eq. (8) on detecting different types of anomalies in the Cora and Books dataset.

5.4 Detection Performance Comparison

5.4.1 GAD-NR shows the superior performance in different types of anomaly detection. In Table 2, we show the results of GAD-NR on the benchmark anomaly detection with baseline models. In Table 4, we present the results in injected contextual, structural and joint-type anomaly detection. From the results, we can observe that GAD-NR outperforms the baseline methods over most of the datasets in detecting benchmark anomaly labels, contextual anomaly labels, and structural + joint-type anomalies.

The key reason behind the performance improvement can be attributed to the entire neighborhood reconstruction around a target node, which includes its self-feature reconstruction, degree reconstruction, and neighbor-feature distribution reconstruction.

The feature-based models like MLPAE perform quite well at detecting contextual anomalies, specifically in the Cora dataset as they put emphasis on self-feature reconstruction. However, MLPAE performs worse to detect joint-type and structural anomalies as they ignore the graph structure. Approaches that only consider the structure information, for example, SCAN performs exceedingly well in detecting structural + joint-type anomalies but they are incapable of doing well in contextual anomaly detection. In the case of the GAE-based models, the performance is more competitive while still worse than GAD-NR in Table 2 and Table 4.

5.4.2 Impacts of different types of reconstruction losses. In Table 2, Table 4 along with the performance of GAD-NR with all three types of reconstruction loss, we have also shown results by removing each part from the loss function of the GAD-NR's decoder, i.e., setting $\lambda_x = 0$, $\lambda_d = 0$ or $\lambda_n = 0$ in Eq. (7). From the results in Table 4, it is clearly visible that without the neighborhood reconstruction part $\lambda_n = 0$, the performance of GAD-NR drops the most in both types of anomaly detection.

From the results of Table 4, we can observe that without the self-feature reconstruction loss ($\lambda_x = 0$), the performance of GAD-NR drops down heavily when detecting contextual anomalies. When detecting the structural + joint-type anomalies, the performance decay of GAD-NR is moderate, which matches the expectation. By removing the loss for degree reconstruction ($\lambda_d = 0$), GAD-NR also suffers from some performance decay. However, such decay is less severe compared to that of removing self-feature reconstruction for detecting contextual anomalies or that of removing neighbors' feature distribution reconstruction for detecting structural anomalies and joint-type anomalies.

5.4.3 Performance Comparison with NWR-GAE [66]. We compare GAD-NR with NWR-GAE [66] in two aspects: performance and running time. For performance, we are adding the benchmark anomaly detection performance comparison between NWR-GAE and our model GAD-NR in Table 3. We can observe that the performance of GAD-NR is significantly better than NWR-GAE in all six datasets. NWR-GAE directly tries to match the empirical distribution of neighbor representation, by which NWR-GAE may capture neighbors' features more precisely (also time consuming) but it tends to overfit anomalous behaviors, compared to the Gaussian approximation that GAD-NR adopts. For running time comparison, we also added the comparison between NWR-GAE and our model GAD-NR in Table 3. Optimizing the KL-divergence leads to a running time complexity of $O(d)$ from the neighborhood matching the Hungarian algorithm's running time complexity of $O(d^3)$. Therefore, GAD-NR is far more scalable on a relatively large graph dataset compared with NWR-GAE for detecting anomalies.

5.5 Hyperparameter Analysis

5.5.1 Impacts of tuning λ'_x, λ'_d, and λ'_n. We present the trend of GAD-NR's performance on different types of anomaly detection by varying the weights in Eq. (8) to perform detection in Figure 3. While increasing the weight for self-feature reconstruction λ'_x in Figure 3 left top, we have observed the performance curve of contextual anomaly detection (blue) is very steep. Similar in 3 left bottom, a growing trend has been observed on both contextual and joint-type anomaly detection performance curve (blue and green). The reason is intuitive. With higher weights for self-feature reconstruction, the decoder of GAD-NR tends to assign higher importance to contextual anomalies as well as joint-type anomalies. By varying the weight for degree reconstruction, λ'_d in Figure 3 middle column, the change in performance is not that significant across different types of anomalies. This is because contextual and structural anomalies do not have much change in node degrees. For the joint-type anomalies, where node degrees may provide useful signals for detecting, only checking node degrees is often insufficient to determine an anomaly. This is because a normal node can also have higher degrees. Node degree reconstruction should be paired with neighbors' feature distribution reconstruction together to provide effective anomaly detection. Lastly, in Figure 3 right column, when we vary the weight λ'_n for neighborhood reconstruction, we notice a significant performance gain in joint-type anomaly detection and structural + joint-type anomaly detection, which demonstrates the effectiveness of neighborhood reconstruction via leveraging signals from their neighborhoods.

Dataset	Cora				Reddit			
# Dimensions / Models	32	64	128	256	8	16	32	64
MLPAE	71.07	71.07	71.08	70.51	47.57	52.12	51.79	51.92
GCNAE	71.47	71.52	71.63	70.84	50.88	51.29	51.81	52.10
DOMINANT	84.52	84.77	84.90	76.77	52.93	52.94	52.99	53.04
DONE	84.19	84.29	86.52	78.29	52.39	52.41	55.15	55.86
AdONE	84.01	84.43	84.87	73.42	57.78	53.64	54.85	55.14
GAAN	74.33	74.15	74.32	76.15	50.21	52.32	52.42	52.79
CONAD	84.54	84.79	84.46	76.15	52.74	52.95	53.03	53.13
GAD-NR	**86.38**	**86.93**	**87.55**	**82.67**	**53.12**	**57.99**	**58.12**	**56.07**

Table 5: Performance comparison of GAD-NR with different latent dimension sizes for detecting benchmark anomalies in Cora and Reddit datasets.

5.5.2 Impact of the latent representation's dimension. In Table 5, we present the performance of GAD-NR on benchmark anomaly detection in the Cora and Reddit datasets by varying the dimension size of hidden representations. From the results, we can observe that the performance of GAD-NR gradually increases as the latent dimension increases for Cora (32 to 128) and Reddit (8 to 32) compared to other GAE-based methods. We think using neighborhood reconstruction is the reason behind the gradual performance improvement of GAD-NR. Other autoencoders can only increase the capability of latent representations by increasing the latent dimension. Instead, GAD-NR can also increase the supervision strength of neighborhood reconstruction by increasing the latent dimension. When the dimension size increases even more e.g. 256 in Cora and 64 in Reddit, the anomaly detection performance of GAD-NR drops. With a higher latent dimension size, the model becomes too much expressive and it can overfit the anomalies. For anomaly detection, we are expected to capture normal behaviors instead of making models memorize all information in the data, especially abnormal behaviors. Therefore, we need to strike a balance between model expressiveness and the proportion of normal information extracted for the best anomaly detection performance.

6 CONCLUSION

In this study, we introduce GAD-NR, for identifying anomalous nodes in graph structures. GAD-NR is based on a graph autoencoder that reconstructs the neighborhood information from node representations generated by a GNN encoder. The reconstruction process encompasses a self-feature representation, degree reconstruction, and the distribution of neighboring node representations, thus allowing the detection of various anomalies including contextual, structural, and joint-type anomalies. Experimental results on six real-world datasets demonstrate the effectiveness of neighborhood reconstruction in identifying different types of anomalies. GAD-NR outperforms state-of-the-art GAD baselines in five out of the six datasets in benchmark evaluations. Additionally, GAD-NR provides flexibility and potential for detecting different types of anomalies through the combination of different types of reconstruction loss with varying weights. GAD-NR also shows the robustness of the selection of weights to detect real-world anomalies.

7 ACKNOWLEDGEMENT

Amit Roy and Pan Li were supported partially by the Sony Award and the NSF grant IIS-2239565. The authors would like to greatly thank Prof. Bruno Ribeiro, Prof Ruqi Zhang, Prof. Sharon (Yixuan) Li and anonymous reviewers for their insightful suggestions to improve the paper.

REFERENCES

[1] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *SIGMOD*, 2001.

[2] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph-based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 2015.

[4] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. Fast and reliable anomaly detection in categorical data. In *CIKM*, 2012.

[5] Sambaran Bandyopadhyay, N Lokesh, and M Narasimha Murty. Outlier aware network embedding for attributed networks. In *AAAI*, 2019.

[6] Sambaran Bandyopadhyay, N Lokesh, and M Narasimha Murty. Outlier aware network embedding for attributed networks. In *AAAI*, 2019.

[7] Sambaran Bandyopadhyay, Saley Vishal Vivek, and MN Murty. Outlier resistant unsupervised deep architectures for attributed network embedding. In *WSDM*, 2020.

[8] Sambaran Bandyopadhyay, Saley Vishal Vivek, and MN Murty. Outlier resistant unsupervised deep architectures for attributed network embedding. In *WSDM*, 2020.

[9] Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *AAAI*, 2018.

[10] Tianyi Chen and Charalampos Tsourakakis. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *KDD*, 2022.

[11] Zhenxing Chen, Bo Liu, Meiqing Wang, Peng Dai, Jun Lv, and Liefeng Bo. Generative adversarial attributed network anomaly detection. In *CIKM*, 2020.

[12] Javier E Contreras-Reyes and Reinaldo B Arellano-Valle. Kullback–leibler divergence measure for multivariate skew-normal distributions. *Entropy*, 2012.

[13] Benoıt Corsini, Pierre-André Noël, David Vázquez, and Perouz Taslakian. Self-supervised anomaly detection in static attributed graphs. *arxiv*, 2021.

[14] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *NeurIPS*, 2020.

[15] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *SDM*. SIAM, 2019.

[16] Kaize Ding, Jundong Li, and Huan Liu. Interactive anomaly detection on attributed networks. In *WSDM*, 2019.

[17] Qi Ding, Natallia Katenka, Paul Barford, Eric Kolaczyk, and Mark Crovella. Intrusion as (anti) social communication: characterization and detection. In *KDD*, 2012.

[18] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*, 2020.

[19] Andrew Elliott, Mihai Cucuringu, Milton Martinez Luaces, Paul Reidy, and Gesine Reinert. Anomaly detection in networks with application to financial transaction networks. *arXiv preprint arXiv:1901.00402*, 2019.

[20] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP*. IEEE, 2020.

[21] Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. Botrgcn: Twitter bot detection with relational graph convolutional networks. In *ASONAM*, 2021.

[22] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*. PMLR, 2017.

[23] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[24] Leonardo Gutiérrez-Gómez, Alexandre Bovet, and Jean-Charles Delvenne. Multi-scale anomaly detection on attributed networks. In *AAAI*, 2020.

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 2017.

[26] Nicholas A Heard, David J Weston, Kiriaki Platanioti, and David J Hand. Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 2010.

[27] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 2006.

[28] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *KDD*, 2016.

[29] Xia Hu, Jiliang Tang, and Huan Liu. Online social spammer detection. In *AAAI*, 2014.

[30] Tianjin Huang, Yulong Pei, Vlado Menkovski, and Mykola Pechenizkiy. Hop-count based self-supervised anomaly detection on attributed networks. In *ECML PKDD*. Springer, 2023.

[31] Tsuyoshi Idé and Hisashi Kashima. Eigenspace-based anomaly detection in computer systems. In *KDD*, 2004.

[32] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Random walk based fake account detection in online social networks. In *DSN*. IEEE, 2017.

[33] Ming Jin, Yixin Liu, Yu Zheng, Lianhua Chi, Yuan-Fang Li, and Shirui Pan. Anemone: graph anomaly detection with multi-scale contrastive learning. In *CIKM*, 2021.

[34] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 1987.

[35] Asif Karim, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, and Mamoun Alazab. A comprehensive survey for intelligent spam email detection. *IEEE Access*, 2019.

[36] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

[37] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

[38] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[39] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. *NeurIPS*, 2007.

[40] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. Radar: Residual analysis for anomaly detection in attributed networks. In *IJCAI*, 2017.

[41] Yuening Li, Xiao Huang, Jundong Li, Mengnan Du, and Na Zou. Specae: Spectral autoencoder for anomaly detection in attributed networks. In *CIKM*, 2019.

[42] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *TKDD*, 2012.

[43] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. In *NeurIPS Datasets and Benchmarks Track*, 2022.

[44] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. Pygod: A python library for graph outlier detection. *arXiv preprint arXiv:2204.12095*, 2022.

[45] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *TNNLS*, 2021.

[46] Zhiyuan Liu, Chunjie Cao, and Jingzhang Sun. Mul-gad: a semi-supervised graph anomaly detection framework via aggregating multi-view information. *arXiv preprint arXiv:2212.05478*, 2022.

[47] Xuexiong Luo, Jia Wu, Amin Beheshti, Jian Yang, Xiankun Zhang, Yuan Wang, and Shan Xue. Comga: Community-aware attributed graph anomaly detection. In *WSDM*, 2022.

[48] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *TKDE*, 2021.

[49] Asep Maulana and Martin Atzmueller. Centrality-based anomaly detection on multi-layer networks using many-objective optimization. In *CoDIT*. IEEE, 2020.

[50] Benjamin Miller, Nadya Bliss, and Patrick Wolfe. Subgraph detection using eigenvector l1 norms. *NeurIPS*, 23, 2010.

[51] Peter Morales, Rajmonda Sulo Caceres, and Tina Eliassi-Rad. Selective network discovery via deep reinforcement learning on embedded spaces. *Applied Network Science*, 2021.

[52] Weisen Pan, Jian Li, Lisa Gao, Liexiang Yue, Yan Yang, Lingli Deng, and Chao Deng. Semantic graph neural network: A conversion from spam email classification to graph classification. *Scientific Programming*, 2022, 2022.

[53] Yulong Pei, Fang Lyu, Werner Van Ipenburg, and Mykola Pechenizkiy. Subgraph anomaly detection in financial transaction networks. In *ICAIF*, 2020.

[54] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, and Qinghua Zheng. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, 2018.

[55] Zhen Peng, Minnan Luo, Jundong Li, Luguo Xue, and Qinghua Zheng. A deep multi-view framework for anomaly detection on attributed networks. *IEEE TKDE*, 2020.

[56] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *KDD*, 2014.

[57] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[58] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA*, 2014.

[59] David Savage, Xiuzhen Zhang, Xinghuo Yu, Pauline Chou, and Qingmai Wang. Anomaly detection in online social networks. *Social networks*, 2014.

[60] HB Mihiri Shashikala, Roy George, and Khalil A Shujaee. Outlier detection in network data using the betweenness centrality. In *SoutheastCon 2015*. IEEE, 2015.

[61] Xiang-Rong Sheng, De-Chuan Zhan, Su Lu, and Yuan Jiang. Multi-view anomaly detection: Neighborhood in locality matters. In *AAAI*, 2019.

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

[63] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, 2015.

[64] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *NeurIPS*, 2023.

[65] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In *ICML*, 2022.

[66] Mingyue Tang, Carl Yang, and Pan Li. Graph auto-encoder via neighborhood wasserstein reconstruction. *ICLR*, 2022.
[67] Hanghang Tong and Ching-Yung Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *SDM*. SIAM, 2011.
[68] Véronique Van Vlasselaer, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Gotcha! network-based fraud detection for social security fraud. *Management Science*, 2017.
[69] Andrew Z Wang, Rex Ying, Pan Li, Nikhil Rao, Karthik Subbian, and Jure Leskovec. Bipartite dynamic representations for abuse detection. In *KDD*, 2021.
[70] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *ICDM*. IEEE, 2019.
[71] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xiong. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *WWW*, 2019.
[72] Qi Wang, Weiliang Zhao, Jian Yang, Jia Wu, Chuan Zhou, and Qianli Xing. Atnetrust: Attributed trust network embedding for trust prediction in online social networks. In *ICDM*. IEEE, 2020.
[73] Jia Wu, Shirui Pan, Xingquan Zhu, and Zhihua Cai. Boosting for multi-graph classification. *IEEE Transactions on Cybernetics*, 2014.
[74] Jia Wu, Xingquan Zhu, Chengqi Zhang, and Zhihua Cai. Multi-instance multi-graph dual embedding learning. In *ICDM*. IEEE, 2013.
[75] Cao Xiao, David Mandell Freeman, and Theodore Hwa. Detecting clusters of fake accounts in online social networks. In *AISec*, 2015.
[76] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.
[77] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. Scan: a structural clustering algorithm for networks. In *SIGKDD*, 2007.
[78] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. Contrastive attributed network anomaly detection with data augmentation. In *PAKDD*. Springer, 2022.
[79] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *ECML PKDD*. Springer, 2015.
[80] Rose Yu, Huida Qiu, Zhen Wen, ChingYung Lin, and Yan Liu. A survey on social media anomaly detection. *ACM SIGKDD Explorations Newsletter*, 2016.
[81] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *SIGIR*, 2020.
[82] Shuang Zhou, Xiao Huang, Ninghao Liu, Huachi Zhou, Fu-Lai Chung, and Long-Kai Huang. Improving generalizability of graph anomaly detection models via data augmentation. *IEEE TKDE*, 2023.
[83] Mengxiao Zhu and Haogang Zhu. Mixedad: A scalable algorithm for detecting mixed anomalies in attributed graphs. In *AAAI*, 2020.