Anomaly Detection in Networks via Score-Based Generative Models

Dmitrii Gavrilev [1] **Evgeny Burnaev** [1 2]

Abstract

Node outlier detection in attributed graphs is a challenging problem for which there is no method that would work well across different datasets. Motivated by the state-of-the-art results of score-based models in graph generative modeling, we propose to incorporate them into the aforementioned problem. Our method achieves competitive results on small-scale graphs. We provide an empirical analysis of the Dirichlet energy, and show that generative models might struggle to accurately reconstruct it.

1. Introduction

Graphs are a natural structure to represent various kinds of data, such as social networks, molecules, the Internet, and infrastructure networks, to name but a few. They describe the connectivity (relations) between objects. Analyzing networks might be crucial for both research and industry, e.g., for fraud detection. For instance, anomalous nodes might reveal fraudsters in a network of transactions, potentially preventing a significant loss of money (Ma et al., 2021) or detecting false product reviews that could mislead customers (Kumar et al., 2018). Another important scenario is recognizing the Out-Of-Distribution (OOD) nodes in a graph, on which a discriminative model may yield unreliable predictions (Wu et al., 2023).

A graph G can be represented by the sets of nodes (vertices) V and edges E. One of the ways to describe a graph is to build the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where n is the number of nodes. Real-world networks are often attributed graphs, meaning that the nodes and edges might additionally have attributes described as vectors. In this paper, we use the notions of 'attributes' and 'features' interchangeably. Anomalous Node Detection (ANOS ND) is the problem of recognizing the nodes in a graph that deviate dramatically from the others (Ma et al., 2021). That is, the goal is to rank the nodes by the degree of abnormality. Due to the cost of labeling training datasets, unsupervised methods are preferred. However, to our knowledge, most of them fail to solve ANOS ND (Liu et al., 2022b). Hence, devising an unsupervised method that would work well across a wide variety of graphs remains challenging.

Recently, score-based generative modeling (or diffusion-based modeling) has been getting close attention due to its expressive generation. It has been incorporated in various modalities, including images (Dhariwal & Nichol, 2021), audio (Chen et al., 2020a), video (Ho et al., 2022), text (Reid et al., 2022), and graphs (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2022). In our work, we leverage score-based graph generative models to detect anomalous nodes in a given attributed network. The behavior of a node can be characterized by its neighborhood, i.e., ego-graph. Formally, an ego-graph of a node v is the induced subgraph of v and its k-hop neighbors (Freeman, 1982). Our key idea is to view a network as a *collection of ego-graphs*. This view allows us to learn the probability distribution induced by a network with score-based generative models. In the context of anomaly detection on networks, our contribution is twofold:

- Learning the distribution of ego-graphs with score-based modeling;
- Introducing measures of abnormality based on the reconstruction error of *attributed* ego-graphs.

This paper is organized as follows. In Appendix A, we review unsupervised methods for ANOS ND and the recent advancements in graph generation. In Section 2, we briefly describe the training procedure of GDSS *Graph Diffusion via the System of Stochastic Differential Equations* (GDSS (Jo et al., 2022)). Next, in Section 3, we present our method of assigning anomaly scores. Section 4 describes the experimental setup, with Appendices B-G supplementing it. Section 5 concludes with a discussion of the results and provides insights regarding the limitations of our methods and directions for improvement.

[1] Skolkovo Institute of Science and Technology, Moscow, Russia [2] Artificial Intelligence Research Institute, Moscow, Russia. Correspondence to: Dmitrii Gavrilev <dmitrygavrilyev@gmail.com>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling

2. Background: Training GDSS

Let $\mathbf{G}_0 = (\mathbf{X}_0, \mathbf{A}_0) \in \mathbb{R}^{N \times F} \times \mathbb{R}^{N \times N}$ be an attributed graph with node feature matrix $\mathbf{X}_0$ and adjacency matrix $\mathbf{A}_0$, where N and F denote the number of nodes and features, respectively. A graph $\mathbf{G}_0$ is an arbitrary ego-graph drawn from the distribution p_{data} we want to learn. Forward diffusion of GDSS is a continuous process that destroys the graph structure and its properties. It is defined by the following Itô SDE:

$$\mathrm{d}\mathbf{G}_t = \mathbf{f}_t(\mathbf{G_t})\,\mathrm{d}t + g_t\,\mathrm{d}\mathbf{w}_t\,, \tag{1}$$

where $\mathbf{f}_t$ and g_t are the drift and diffusion functions, respectively, and $\mathbf{w}_t$ is the Wiener process. Note that $\mathbf{G}_t$, $\mathbf{f}_t$, g_t and $\mathbf{w}_t$ are all functions of time t, which is written as a subscript for brevity. This process spans over a time horizon $[0, T]$. Therefore, $\mathbf{G}_0 \sim p_{\text{data}}$ denotes the original graph, whereas $\mathbf{G}_t \sim p_{0t}(\mathbf{G}_t|\mathbf{G}_0)$ indicates its terminal noisy version. Note that $\mathbf{G}_t$ at $t > 0$ is not a sparse graph since forward diffusion destroys the sparsity of the adjacency matrix.

Let the drift function be separable into linear attribute and adjacency components:

$$\mathbf{f}_t(\mathbf{G}_t) = (\mathbf{f}_{1,t}(\mathbf{X}_t), \mathbf{f}_{2,t}(\mathbf{A}_t))\,. \tag{2}$$

The choice of the linear drift terms allows us to factorize the transition kernel $p_{0t}(\mathbf{G}_t|\mathbf{G}_0)$:

$$p_{0t}(\mathbf{G}_t|\mathbf{G}_0) = p_{0t}(\mathbf{X}_t|\mathbf{X}_0)p_{0t}(\mathbf{A}_t|\mathbf{A}_0). \tag{3}$$

Moreover, sampling from $p_{0t}(\mathbf{X}_t|\mathbf{X}_0)$ and $p_{0t}(\mathbf{A}_t|\mathbf{A}_0)$ is fast because they are Gaussian, each with a known mean and covariance (see Sections 5.5 and 6.1 in (Särkkä & Solin, 2019) for more details). Thus, simulating the entire forward process is not required. When the SDE is either Variance Preserving (VP), Variance Exploding, or sub-VP, its transition kernel takes the following functional form (Song et al., 2020):

$$p_{0t}(\mathbf{X}_t|\mathbf{X}_0) = \mathcal{N}\left(\mathbf{X}_t; m_t\mathbf{X}_0, \sigma_t^2\mathbf{I}\right),$$

where m_t is a scalar function of time, to which we refer to as the *signal decay factor*. Similarly, the SDE for $\mathbf{A}_t$ has the same functional form.

The reverse diffusion of GDSS is defined by the system of SDEs:

$$\begin{cases} \mathrm{d}\mathbf{X}_t = \left[\mathbf{f}_{1,t}(\mathbf{X}_t) - g_{1,t}^2\nabla_{\mathbf{X}_t}\log p_t(\mathbf{X}_t, \mathbf{A}_t)\right]\mathrm{d}\bar{t} + g_{1,t}\,\mathrm{d}\overline{\mathbf{w}}_1 \\ \mathrm{d}\mathbf{A}_t = \left[\mathbf{f}_{2,t}(\mathbf{A}_t) - g_{2,t}^2\nabla_{\mathbf{A}_t}\log p_t(\mathbf{X}_t, \mathbf{A}_t)\right]\mathrm{d}\bar{t} + g_{2,t}\,\mathrm{d}\overline{\mathbf{w}}_2 \end{cases}, \tag{4}$$

where $\mathrm{d}\overline{\mathbf{w}_1}$, $\mathrm{d}\overline{\mathbf{w}_2}$ and $\mathrm{d}\bar{t}$ correspond to the time reversal.

The partial scores $\nabla_{\mathbf{X}_t}\log p_t(\mathbf{X}_t, \mathbf{A}_t)$ and $\nabla_{\mathbf{A}_t}\log p_t(\mathbf{X}_t, \mathbf{A}_t)$ are modeled with neural networks $\mathbf{s}_{\theta,t}$ and $\mathbf{s}_{\phi,t}$, respectively, where θ and ϕ are the sets of parameters. The training objectives for these networks are the following:

$$\begin{aligned} &\min_\theta \mathbb{E}_t\left\{\lambda_1(t)\mathbb{E}_{\mathbf{G}_0}\mathbb{E}_{\mathbf{G}_t|\mathbf{G}_0}\|\mathbf{s}_{\theta,t}(\mathbf{G}_t) - \nabla_{\mathbf{X}_t}\log p_{0t}(\mathbf{X}_t|\mathbf{X}_0)\|_2^2\right\} \\ &\min_\phi \mathbb{E}_t\left\{\lambda_2(t)\mathbb{E}_{\mathbf{G}_0}\mathbb{E}_{\mathbf{G}_t|\mathbf{G}_0}\|\mathbf{s}_{\phi,t}(\mathbf{G}_t) - \nabla_{\mathbf{A}_t}\log p_{0t}(\mathbf{A}_t|\mathbf{A}_0)\|_2^2\right\} \end{aligned} \tag{5}$$

Positive weights $\lambda_1(t)$ and $\lambda_2(t)$ indicate the strength of score matching at time t.

The effect of hubs In our setting, $\mathbf{G}_0$ denotes an uncorrupted ego-graph. Therefore, to estimate the expectation $\mathbb{E}_{\mathbf{G}_0}$ in Eq. (5), we sample a random node v from $\mathcal{G}$ and then build an ego-graph $\mathbf{G}_0 = \mathbf{G}_0(v)$. Since the score network for adjacency matrix $s_\phi(\mathbf{G}_t) \in \mathbb{R}^{N \times N}$, both training and inference scale quadratically w.r.t. the number of nodes. Hence, training GDSS might be computationally intractable for large graphs. Real-world networks may contain the so-called ”hubs”, i.e. nodes with a significant number of links (Barabási & Albert, 1999; Barabási & Bonabeau, 2003). Consequently, even the 1-hop ego-graph of a hub can hinder mini-batch training. To alleviate this issue, we propose to sample subgraphs in which the number of nodes does not exceed some predefined value M. Specifically, if a sampled ego-graph has a number of nodes that exceeds M, we simply truncate it by subsampling a subgraph of size M. Note that in this subsampling procedure, we make sure that the subgraph contains the central node of the original ego-graph. For brevity, we do not differentiate between the original ego-graph and its subgraph.

3. Ego-graph Reconstruction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ be an unweighted attributed network with the vertex set $\mathcal{V}$, edge set $\mathcal{E}$, and node attribute matrix $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}| \times F}$. The goal is to rank the nodes from $\mathcal{V}$ such that anomalous nodes are placed higher than normal nodes. A common strategy for solving node outlier detection is to design an unsupervised scoring function $\mathrm{score}(\cdot) : \mathcal{V} \to \mathbb{R}$ that defines the node ranking (Ma et al., 2021). As a convention, we consider positive examples to be outliers. Consequently, larger scores should reflect higher degrees of abnormalities.

Each node v of $\mathcal{G}$ induces a k-hop ego-graph $\mathbf{G}(v)$. We assume that there is a hidden underlying distribution of ego-graphs p_{data} from which the set of observed samples $\{\mathbf{G}(v) \,|\, v \in \mathcal{V}\}$ is drawn. To solve ANOS ND, we propose to learn this distribution through GDSS (Song et al., 2020). We assign anomaly scores using a dissimilarity between the original and reconstructed ego-graphs with score-based generative models. Our key assumption is that the generator reconstructs inlier ego-graphs more accurately than outliers.

After learning the distribution of ego-graphs, we can calculate the reconstruction error for each node. Given node v and time τ, the reconstruction operator acts as follows:

1. Run the forward diffusion (Eq. (1)) on $\mathbf{G}(v)$ until time τ, resulting in $\mathbf{G}_\tau(v)$
2. Solve the reverse diffusion (Eq. (4)) with an initial condition on $\mathbf{G}_\tau(v)$, resulting in $\hat{\mathbf{G}}(v,\tau)$

We combine the ideas of sampling with different levels of noise (Graham et al., 2022) and sampling multiple noisy versions of the same example (Liu et al., 2022c). The ego-graphs are reconstructed several times with different values of $\tau = \tau_1, \ldots, \tau_K$ distributed uniformly in $[0, T]$. At each time τ_i, we sample S noisy versions of the ego-graphs from $p_{0\tau_i}$. Let us denote $\mathbf{G}_\tau^{(1)}(v), \mathbf{G}_\tau^{(2)}(v), \ldots, \mathbf{G}_\tau^{(S)}(v)$ the independently corrupted ego-graphs centered around node v at time τ. Further, we denote $\hat{\mathbf{G}}^{(1)}(v,\tau), \hat{\mathbf{G}}^{(2)}(v,\tau), \ldots, \hat{\mathbf{G}}^{(S)}(v,\tau)$ the corresponding reconstructions. The system in Eq. (4) can be solved numerically with the Euler-Maruyama or Predictor-Corrector methods (Song et al., 2020). In addition, the authors of GDSS present a novel solver, Symmetric Splitting for System of SDEs (S4) (Jo et al., 2022).

Time τ is associated with the level of noise: the higher values of τ correspond to the lower values of the Signal-to-Noise Ratio (SNR). Given the variance of the perturbation kernel σ_t^2 and the signal decay factor m_t (see Eq. (2)), SNR at time τ (Kingma et al., 2021) can be defined as

$$\mathrm{SNR}(\tau) = \frac{m_\tau^2}{\sigma_\tau^2}. \tag{6}$$

In the limit $\tau \to \infty$, the original signal is completely diminished, and the reconstruction operator acts blindly. Thus, we propose to reweight the errors at different noise scales by $\sqrt{\mathrm{SNR}(\tau)}$.

Given a dissimilarity measure $d(\cdot,\cdot)$ on graphs of the same size and a time penalty function $\gamma(\cdot)$, we define the anomaly score for node v as

$$\mathrm{score}(v) = \sum_{i=1}^{K}\sum_{j=1}^{S}\left(\gamma(\tau_i)\cdot d(\mathbf{G}^{(j)}(v,\tau_i), \hat{\mathbf{G}}(v))\right). \tag{7}$$

In this work, we set $\gamma(\tau)$ as either $\mathrm{SNR}(\tau)$ or 1 (no weighting). As for ego-graph dissimilarity, we propose two different ways of measuring it: 1) as a convex combination of **matrix** distances; 2) as the difference in normalized **energies**.

Matrix distance One of the common ways to define a dissimilarity measure for graphs is to sum the distances between adjacency and feature matrices (Ding et al., 2019):

$$d(\mathbf{G}, \hat{\mathbf{G}}) = (1-\alpha)\cdot\frac{\|\mathbf{A}-\hat{\mathbf{A}}\|_F}{N^2} + \alpha\cdot\frac{\|\mathbf{X}-\hat{\mathbf{X}}\|_F}{N\cdot F}, \tag{8}$$

where $\alpha \in [0,1]$ is a hyperparameter, N and F are the numbers of nodes and features, respectively. In addition, we normalize matrices by their size. Normalizing the errors in adjacency matrices by their dimensionality helps to deal with the bias towards larger ego-graphs. Moreover, the normalization of the feature matrix error allows us to choose the weight α across different datasets more consistently since F depends on the dataset.

Shift in energy Let $\mathbf{D}$ be the diagonal matrix of node degrees and $\mathbf{L}$ be the normalized Laplacian:

$$\mathbf{L} = \mathbf{D}^{\dagger/2}(\mathbf{D}-\mathbf{A})\mathbf{D}^{\dagger/2}. \tag{9}$$

Note that instead of taking the exact inverse square root of $\mathbf{D}$, we operate with its pseudoinverse square root. This allows us to normalize the Laplacian even if the reconstructed ego-graphs contain isolated nodes. If a graph is directed, then we symmetrize its Laplacian. Both features and structure can be incorporated into a single functional, the Dirichlet energy of a graph, which is defined as the variation of features along the edges (Cai & Wang, 2020):

$$\mathcal{E}(\mathbf{X},\mathbf{L}) = \sum_{(i,j)\in E}\left\|\frac{\mathbf{X}_i}{\sqrt{D_{ii}}} - \frac{\mathbf{X}_j}{\sqrt{D_{jj}}}\right\|^2 = \operatorname{Tr}\mathbf{X}^\intercal\mathbf{L}\mathbf{X}. \tag{10}$$

It can be interpreted as a measure of feature smoothness, with lower values of the energy indicating that the adjacent nodes have similar features. In general, the Dirichlet energy is unbounded above. Following (Di Giovanni et al., 2022), we normalize the energy by the squared Frobenius norm of features, which yields a quantity bounded by the Laplacian spectral radius $\rho_\mathbf{L}$:

$$0 \le \frac{\mathcal{E}(\mathbf{X},\mathbf{L})}{\|\mathbf{X}\|^2} \le \rho_\mathbf{L} \le 2. \tag{11}$$

Contrary to the previous studies (Cai & Wang, 2020; Zhou et al., 2021; Di Giovanni et al., 2022), we view energy as a functional of both features and the Laplacian. Bounding the energy helps quantify the shift in energy, which we define through the absolute difference between reconstructed and original energies:

$$d(\mathbf{G},\hat{\mathbf{G}}) = \left|\frac{\mathcal{E}(\mathbf{X},\mathbf{L})}{\|\mathbf{X}\|^2} - \frac{\mathcal{E}(\hat{\mathbf{X}},\hat{\mathbf{L}})}{\|\hat{\mathbf{X}}\|^2}\right|. \tag{12}$$

A large gap between energies indicates a drastic change in how the node features align with each other as well as structural changes. Thus, a shift in energy can serve as a dissimilarity measure.

4. Experiments

To assess the quality of our method, we follow the evaluation protocol from the BOND benchmark (Liu et al., 2022b).

Table 1. ROC-AUC (%) on datasets with organic outliers. The best average results are written in **bold**, and the best maximum results are underlined. TLE and OOM_C indicate that the method exceeded the time limit of 24 hours and failed to fit in VRAM, respectively.

Algorithm	**Weibo**	**Reddit**	**Disney**	**Books**	**Enron**	**DGraph**
LOF	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)	TLE
IF	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)	**60.9** ± 0.7(62.0)
MLPAE	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)	37.0 ± 1.9 (41.3)
SCAN	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)	TLE
Radar	**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)	OOM_C
ANOMALOUS	**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)	OOM_C
GCNAE	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)	40.9 ± 0.5 (42.2)
DOMINANT	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)	OOM_C
DONE	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)	OOM_C
AdONE	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)	OOM_C
AnomalyDAE	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)	OOM_C
GAAN	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)	OOM_C
GUIDE	OOM_C	OOM_C	38.8 ± 8.9 (52.5)	48.4 ± 4.6 (63.5)	OOM_C	OOM_C
CONAD	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)	34.7 ± 1.2 (36.5)
Rec	74.5 ± 12.6 (88.6)	44.4 ± 0.4 (45.1)	**65.0** ± 11.2 (79.8)	56.9 ± 2.7 (62.1)	44.0 ± 4.0 (51.4)	TLE
Rec (unweighted)	74.4 ± 12.4 (88.2)	44.5 ± 0.4 (45.2)	63.1 ± 12.9 (78.1)	57.1 ± 2.8 (62.7)	44.0 ± 4.2 (51.4)	TLE
Energy	51.3 ± 11.0 (64.8)	55.1 ± 0.8 (56.7)	56.7 ± 5.7 (67.0)	52.4 ± 3.8 (59.2)	36.5 ± 5.6 (48.2)	TLE
Energy (unweighted)	51.9 ± 11.1 (67.8)	55.0 ± 0.9 (56.8)	58.4 ± 5.9 (68.2)	52.7 ± 3.0 (57.5)	35.5 ± 5.0 (46.9)	TLE

This benchmark tests 14 different approaches, ranging from matrix factorization to deep neural networks. For fair comparison, tuning of hyperparameters is performed on the shared grid. BOND evaluates the algorithms on real-world networks that include organic and synthetic anomalies. In this work, we evaluate our methods only on graphs with organic outliers. A detailed description of datasets as well as their statistics is provided in Appendix B. In the preprocessing step, we standardize the node feature matrices such that each feature has a unit standard deviation. This allows us to use the same level of noise for each feature during forward diffusion.

If possible, the models from the BOND benchmark share the same grid of hyperparameters. On each dataset, performance is evaluated 20 times. At the beginning of a trial, a model is built with randomly drawn hyperparameters. To solve the reverse diffusion SDEs, we use the Euler-Maruyama solver with $\lfloor 100 \cdot \frac{\tau}{T} \rfloor$ steps. We set the number of reconstruction levels $K = 4$ and the number of samples per level $S = 3$. More details regarding the architecture and hyperparameters can be found in Appendix C. In Appendix E, we motivate our choice of SDE solver.

Tables 1, 5, and 6 show a comparison of our methods and the baselines in terms of metrics (see Appendix D for the description of metrics and additional results). The baseline results are taken from BOND. Our results are shown in the bottom rows (in **bold**). *Rec* stands for reconstruction-based detection with matrix distance, whereas *Energy* corresponds to setting a shift in energy as a dissimilarity measure. By default, we assume that the scores are weighted with the SNR time penalties. We first write the average metrics, the standard deviation, and the maximum in brackets. Ego-graphs are visualized in Appendix G.

Energy reconstruction Further, we investigate how well GDSS reconstructs the energy. Figure 2 shows the original and reconstructed normalized energies from different noise levels. The energy values are collected using all 20 checkpoints. As can be seen from the histograms, GDSS tends to generate ego-graphs with low energies. This effect is more visible in Figure 3, where we plotted histograms of the signed shift in energy $\frac{\mathcal{E}(\mathbf{X},\mathbf{L})}{\|\mathbf{X}\|^2} - \frac{\mathcal{E}(\hat{\mathbf{X}},\hat{\mathbf{L}})}{\|\hat{\mathbf{X}}\|^2}$. A significant bias towards positive values indicates that the reconstructed ego-graphs are either smoother or sparser than their originals.

5. Discussion

In this work, we present a novel method to tackle node outlier detection in attributed networks. Our approach leverages score-based generative graph models to reconstruct ego-graphs and is agnostic to the particular choice of model. We assign anomaly scores based on a dissimilarity measure between the original and reconstructed ego-graphs. We experiment with two ways of measuring the dissimilarity: 1) combining the norms of the differences between both the adjacency and feature matrices; and 2) calculating the absolute shift in normalized Dirichlet energies. The former measure shows the best results on Disney dataset, whereas the rest of the benchmarked methods completely fail. However, it shows results that are poor on two larger datasets, Reddit and Enron, and moderate on the others. The latter measure is consistently worse than the matrix distance, except on Reddit, whose energy distribution forms a narrow band located on smaller values, as opposed to the other graphs. Analyzing the shift in energy might be helpful not only for anomaly detection in networks but also for assessing the quality of graph generative models.

Future directions involve finding an optimal architecture and incorporating more expressive graph convolutions, such as GRAFF (Di Giovanni et al., 2022). Another prospect is guided generation with node positional encodings. Unfolding a graph into a collection of ego-graphs comes with both advantages and shortcomings. It serves as a technique for training score-based models at a local scale, efficiently learning the notion of normality in interactions. Nevertheless, information from neighborhoods may not be sufficient. Employing positional encodings might alleviate this issue by capturing subtle higher-order interactions. They include, but are not limited to, Laplacian PE (Dwivedi et al., 2020), SignNet (Lim et al., 2022), and geodesics (Rampášek et al., 2022).

Acknowledgements

This work was supported by Ministry of Science and Higher Education grant No. 075-10-2021-068.

References

Akoglu, L., McGlohon, M., and Faloutsos, C. Oddball: Spotting anomalies in weighted graphs. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 410–421. Springer, 2010.

Bandyopadhyay, S., Vivek, S. V., and Murty, M. Outlier resistant unsupervised deep architectures for attributed network embedding. In *Proceedings of the 13th international conference on web search and data mining*, pp. 25–33, 2020.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Barabási, A.-L. and Bonabeau, E. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Cai, C. and Wang, Y. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020a.

Chen, Z., Liu, B., Wang, M., Dai, P., Lv, J., and Bo, L. Generative adversarial attributed network anomaly detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1989–1992, 2020b.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*, 2022.

Ding, K., Li, J., Bhanushali, R., and Liu, H. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 594–602. SIAM, 2019.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. 2020.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Freeman, L. C. Centered graphs and the structure of ego networks. *Mathematical Social Sciences*, 3(3):291–304, 1982.

Fruchterman, T. M. and Reingold, E. M. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

Graham, M. S., Pinaya, W. H., Tudosiu, P.-D., Nachev, P., Ourselin, S., and Cardoso, M. J. Denoising diffusion models for out-of-distribution detection. *arXiv preprint arXiv:2211.07740*, 2022.

Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J. (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

Huang, X., Yang, Y., Wang, Y., Wang, C., Zhang, Z., Xu, J., Chen, L., and Vazirgiannis, M. Dgraph: A large-scale financial dataset for graph anomaly detection. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. *arXiv preprint arXiv:2202.02514*, 2022.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 333–341, 2018.

Kumar, S., Zhang, X., and Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1269–1278, 2019.

Leskovec, J., Adamic, L. A., and Huberman, B. A. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007.

Li, J., Dani, H., Hu, X., and Liu, H. Radar: Residual analysis for anomaly detection in attributed networks. In *IJCAI*, volume 17, pp. 2152–2158, 2017.

Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H., and Jegelka, S. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.

Liu, J., Kumar, A., Ba, J., Kiros, J., and Swersky, K. Graph normalizing flows, 2019. URL https://arxiv.org/abs/1905.13177.

Liu, K., Dou, Y., Zhao, Y., Ding, X., Hu, X., Zhang, R., Ding, K., Chen, C., Peng, H., Shu, K., Chen, G. H., Jia, Z., and Yu, P. S. Pygod: A python library for graph outlier detection. *arXiv preprint arXiv:2204.12095*, 2022a.

Liu, K., Dou, Y., Zhao, Y., Ding, X., Hu, X., Zhang, R., Ding, K., Chen, C., Peng, H., Shu, K., Sun, L., Li, J., Chen, G. H., Jia, Z., and Yu, P. S. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *arXiv preprint arXiv:2206.10071*, 2022b.

Liu, L., Ren, Y., Cheng, X., and Zhao, Z. Diffusion denoising process for perceptron bias in out-of-distribution detection. *arXiv preprint arXiv:2211.11255*, 2022c.

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., Xiong, H., and Akoglu, L. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Metsis, V., Androutsopoulos, I., and Paliouras, G. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pp. 28–69. Mountain View, CA, 2006.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling, 2020. URL https://arxiv.org/abs/2003.00638.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peng, Z., Luo, M., Li, J., Liu, H., Zheng, Q., et al. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, pp. 3513–3519, 2018.

Pennebaker, J. W., Francis, M. E., and Booth, R. J. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

Perozzi, B. and Akoglu, L. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 207–215. SIAM, 2016.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Reid, M., Hellendoorn, V. J., and Neubig, G. Diffuser: Discrete diffusion via edit-based reconstruction. *arXiv preprint arXiv:2210.16886*, 2022.

Sakurada, M. and Yairi, T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pp. 4–11, 2014.

Sánchez, P. I., Müller, E., Laforet, F., Keller, F., and Böhm, K. Statistical selection of congruent subspaces for mining attributed graphs. In *2013 IEEE 13th international conference on data mining*, pp. 647–656. IEEE, 2013.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pp. 412–422. Springer, 2018.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Wang, M. Y. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR workshop on representation learning on graphs and manifolds*, 2019.

Wang, Y., Zhang, J., Guo, S., Yin, H., Li, C., and Chen, H. Decoupling representation learning and classification for gnn-based anomaly detection. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 1239–1248, 2021.

Welling, M. and Kipf, T. N. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

Wu, Q., Chen, Y., Yang, C., and Yan, J. Energy-based out-of-distribution detection for graph neural networks. *arXiv preprint arXiv:2302.02914*, 2023.

Xu, X., Yuruk, N., Feng, Z., and Schweiger, T. A. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 824–833, 2007.

Xu, Z., Huang, X., Zhao, Y., Dong, Y., and Li, J. Contrastive attributed network anomaly detection with data augmentation. In *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part II*, pp. 444–457. Springer, 2022.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models, 2018. URL https://arxiv.org/abs/1802.08773.

Yuan, X., Zhou, N., Yu, S., Huang, H., Chen, Z., and Xia, F. Higher-order structure based anomaly detection on attributed networks. In *2021 IEEE International Conference on Big Data (Big Data)*, pp. 2691–2700. IEEE, 2021.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhao, T., Deng, C., Yu, K., Jiang, T., Wang, D., and Jiang, M. Error-bounded graph anomaly loss for gnns. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1873–1882, 2020.

Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S.-H., and Hu, X. Dirichlet energy constrained learning for deep graph neural networks. *Advances in Neural Information Processing Systems*, 34:21834–21846, 2021.

A. Related Work

A.1. Anomaly Detection

In (Ma et al., 2021), the authors present a survey on graph anomaly detection methods. For ANOS ND, they provide the following taxonomy of anomalies: global, in which node attributes differ from the distribution of attributes; structural, in which the graph's structure is considered; and community anomalies, in which attributes differ within the nodes in the same community. They also review work that assigns anomaly scores to individual nodes based on their contribution to the network reconstruction error. The BOND benchmark (Liu et al., 2022b) investigates the performance of

- graph-agnostic algorithms: LOF (Breunig et al., 2000), IF (Liu et al., 2012), MLPAE (Sakurada & Yairi, 2014)
- classical algorithms: SCAN (Xu et al., 2007), RADAR (Li et al., 2017), ANOMALOUS (Peng et al., 2018)
- deep algorithms: GCNAE (Kipf & Welling, 2016), DOMINANT (Ding et al., 2019), DONE, AdONE (Bandyopadhyay et al., 2020), GAAN (Chen et al., 2020b), GUIDE (Yuan et al., 2021), CONAD (Xu et al., 2022).

For instance, DOMINANT (Ding et al., 2019) maps nodes of a graph into the latent space through a graph convolutional encoder (Welling & Kipf, 2016). Then, the latent representations are passed into two separate decoders that reconstruct the adjacency matrix and node features, respectively. The nodes are scored with the weighted sum of the corresponding reconstruction errors.

Ego-networks Previous work regarding the analysis of ego-networks includes the non-deep learning algorithms OddBall (Akoglu et al., 2010) and AMEN (Perozzi & Akoglu, 2016). The former solves ANOS ND on weighted unattributed graphs by employing outlier detection on the graph statistics of ego-networks. OddBall relies on a set of heavy assumptions as to what can be considered an anomaly, e.g., star-shaped or near-clique ego-networks. Therefore, it might not generalize well to arbitrary graphs. Further, AMEN introduces the normality score for attributed subgraphs. For a given subgraph, it measures how its nodes are similar to each other on some subset of attributes, as well as how they are dissimilar to the nodes from the boundary (a set of nodes that do not belong to the subgraph but have at least one neighbor from it).

A.2. Score-Based Modeling for OOD detection

The idea of using diffusion-based generative models in OOD detection appears in (Liu et al., 2022c; Graham et al., 2022). Particularly, they consider the problem of OOD detection in the image domain. In (Liu et al., 2022c), the authors propose to sample neighbors in the input space for a given image by generating them via a diffusion model, which is pre-trained on the in-distribution samples. The neighborhood is constructed by sampling the corrupted images from the forward transition distribution $q(\mathbf{x}_t|\mathbf{x}_0)$. A hyperparameter $t \in [0, T]$ defines the level of corruption. Then, the neighborhood is passed to a discriminator to extract features (e.g., a ResNet (He et al., 2016) pre-trained on an image classification task for in-distribution data). Finally, they compute the OOD scores as the sum of absolute differences between the features of a given image and its neighbors.

An alternative approach, proposed in (Graham et al., 2022), consists of sampling the corrupted images with different levels of noise, distributed uniformly. The OOD scores are assigned as the combination of the MSE in image space and the LPIPS (Learned Perceptual Image Patch Similarity (Zhang et al., 2018)) in AlexNet's feature space (Krizhevsky, 2014).

A.3. Graph Generation

As for graph generative models, there is a great variety of deep-learning methods. They include sequential modeling through variational auto-encoders (Simonovsky & Komodakis, 2018), recurrent neural networks (You et al., 2018), normalizing flows (Liu et al., 2019) and score-based models (Niu et al., 2020; Jo et al., 2022). However, most of the methods consider only non-attributed graphs.

The idea of leveraging score-based models to generate graphs is first explored in (Niu et al., 2020). The authors propose the process of denoising the adjacency matrix. They inject the permutation invariance of the underlying distribution of adjacency matrices as an inductive bias. It is done by designing a permutation equivariant score network, which can be constructed of message passing layers. In particular, the authors introduce EDP-GNN, a graph neural network architecture that is inspired by image dense prediction networks. It consists of the GNN layers that operate on multi-channel adjacency matrices, which are analogous to the feature maps produced by image convolutional layers.

Further, in (Jo et al., 2022), the authors introduce Graph Diffusion via the System of Stochastic differential equations (GDSS) as an extension of the previous method to attributed graphs. They consider continuous-time diffusion, where both node features and the adjacency matrix are perturbed simultaneously. Forward and reverse diffusion processes are induced by stochastic differential equations (SDEs). Solving the reverse diffusion is especially expensive due to the high dimensionality of the score function. Hence, the authors instead aim to approximate the partial score functions that correspond to the node features and adjacency matrix. This results in a system of SDEs that is equivalent to the original reverse diffusion SDE. To that end, GDSS shows state-of-the-art performance in generating generic graphs and competitive performance in molecule generation.

Another diffusion-based model for graph generation, DiGress (Vignac et al., 2022), considers node and edge features to be discrete and drawn from the categorical distribution. Thus, each node and edge is attributed with exactly one type from the attribute space. The structural information is stored in the edge features by introducing the absence of an edge as a separate edge type. Graphs are perturbed through random transformations of their attributes. The transition matrices describe the probabilities of jumping from one type to another. To sample graphs, one needs to design a tractable prior distribution. A naive approach would be to choose uniform distribution over the attribute types as the prior. However, a noisy graph from this prior is highly dense, for which many denoising steps are required. Instead, the authors propose to use the product of the marginal distributions of node and edge types. As a result, the terminal graphs from the prior are much closer to the original ones. DiGress can also be applied to a case of continuous Gaussian noise. Such a modification, ConGress, is similar to GDSS yet models the full score function.

B. Datasets

- **Weibo** (Zhao et al., 2020) is the users-posts-hashtags graph from the social platform of the same name. The users are considered suspicious (anomalies) if their posting frequency resembles that of bots (i.e., every x seconds). BOND employs the **directed** user-user form of the same graph, with hashtags serving as the edges. The node attributes include aggregated information from users' posts, such as the post's location and the text's bag-of-word vectors. Further, the attributes are compressed via dimensionality reduction techniques. Since anomaly labels are derived from timestamps, temporal information is removed from the feature space.
- **Reddit** (Kumar et al., 2019; Wang et al., 2021) is a subset of the user-group (user-subreddit) bipartite graph of the corresponding social media platform. Although both users and groups are considered nodes, there is no label to differentiate between the two. The banned users are assumed to be outliers. The LIWC (Pennebaker et al., 2001) representations of posts are aggregated for each node to construct features.
- **Disney** (Sánchez et al., 2013) is a co-purchase network of movies from Amazon (Leskovec et al., 2007) with manually labeled anomalies. Each movie is attributed with its price, rating, number of reviews, etc.
- **Books** (Sánchez et al., 2013) originates from the Amazon co-purchase network (Leskovec et al., 2007), similar to the Disney dataset. The items labeled with the *amazonfail* tag are considered outliers.
- **Enron** (Sánchez et al., 2013) is an email communication network, where nodes are email addresses and edges are messages. Spam senders are labeled as outliers (Metsis et al., 2006). Each email address is described by statistics such as the average message length and the average number of recipients.
- **DGraph** (Huang et al., 2022) is a financial social network where the nodes represent user accounts. The edge between two users exists if one of them adds the other as an emergency contact. Users with an overdue history are regarded as anomalous. The features include general information such as age, gender, and repayment dates.

C. Implementation Details

Architecture To speed up the evaluation, we use a lightweight variant of GDSS. For score networks $s_{\theta,t}$ and $s_{\phi,t}$, we set the number of GCN and GMH layers to 1. The number of heads for GMH is set to 4. The inputs to GMH are the node feature matrix $\mathbf{X}$ and the adjacency tensor $[\mathbf{A}, \mathbf{A}^2]$. The output of GMH has four channels. The MLP inside the GMH block has two linear layers. GCN and GMH are followed by the channel-mixing MLPs that have three layers. All MLP blocks have the ELU activation (Clevert et al., 2015). We set the form of the forward diffusion SDEs to be Variance Preserving (VP SDE):

$$\mathrm{d}\mathbf{G}_t = -\frac{1}{2}\beta(t)\mathbf{G}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}_t\,, \tag{13}$$

Table 2. The statistics of datasets from BOND. **Ratio** indicates the ratios of outliers in a graph.

Dataset	#Nodes	#Edges	#Features	Avg. Degree	Ratio
Weibo	8,405	407,963	400	48.5	10.3%
Reddit	10,984	168,016	64	15.3	3.3%
Disney	124	335	28	2.7	4.8%
Books	1,418	3,695	21	2.6	2.0%
Enron	13,533	176,987	18	13.1	0.4%
DGraph	3,700,550	4,300,999	17	1.2	0.4%

where $\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min})t$, $\beta_{\min} = 0.1$, $\beta_{\max} = 1$.

Hyperparameters Table 3 presents the pool of hyperparameters common to the algorithms from BOND. *Alpha* denotes the balancing weight for reconstructing the structure and features. The deep learning models are optimized with the Adam algorithm (Kingma & Ba, 2014). Note that Table 3 does not include batch sizes. Due to the high memory consumption of our methods, we set them independently of the BOND benchmark. The corresponding batch sizes for each dataset are shown in Table 4.

Table 3. The grid of hyperparameters shared by the algorithms from BOND.

Hyperparameters	Weibo	Disney	Books	Enron	DGraph	Reddit
learning rate	$[0.1, 0.05, 0.01]$					
weight decay	0.01					
epoch	300				2	300
alpha	$[0.8, 0.5, 0.2]$					
hid. dim.	$[32, 64, 128, 256]$	$[8, 12, 16]$				$[32, 48, 64]$

Table 4. Batch sizes used for training GDSS and inference.

Weibo	Disney	Books	Enron	DGraph	Reddit
2048	full batch	full batch	4096	-	4096

Software All methods are written in Python 3 and use PyTorch for autodifferentiation (Paszke et al., 2017). We employ both DGL (Deep Graph Library (Wang, 2019) and PyG (PyTorch Geometric (Fey & Lenssen, 2019) for graphs, which are popular libraries for training GNNs. For evaluation, we use the code from PyGOD, a library for graph outlier detection (Liu et al., 2022a). Our code and model checkpoints are publicly available at `https://github.com/realfolkcode/GraphDiffusionAnomaly`.

D. Metrics

- **ROC-AUC** assesses the quality of predicted scores by taking into account all possible thresholds that separate negative and positive examples. At each threshold value, the true positive rate (TPR) and the false positive rate (FPR) are calculated. Then, the Receiver Operating Curve is formed by plotting (FPR, TPR) pairs. ROC-AUC is an integral measure defined as the area under the ROC curve. One of the popular interpretations is the probability of a random positive example (anomaly) having a higher score than a random negative example. ROC-AUC equals 1 means that the algorithm perfectly separates anomalies from normal nodes, whereas ROC-AUC equals 0.5 indicates that the model makes random guesses.
- **Average Precision** is calculated as follows:

$$\mathrm{AP} = \sum_{n} \left(R_n - R_{n-1}\right) P_n, \tag{14}$$

where n is the threshold index, P_n and R_n are the precision and recall at the n-th threshold, respectively (Pedregosa et al., 2011). It can be seen as a summarization of the precision-recall curve.

- **Recall@k** indicates the fraction of true outliers among the top-k ranked examples. In the BOND benchmark, k is set as the number of anomalies in a dataset. Hence, Recall@k measures how well the model places the outliers at the top of the list.

Table 5. Average Precision (%) on datasets with organic outliers. The best average results are written in **bold**, and the best maximum results are underlined. TLE and OOM_C indicate that the method exceeded the time limit of 24 hours and failed to fit in VRAM, respectively.

Algorithm	**Weibo**	**Reddit**	**Disney**	**Books**	**Enron**	**DGraph**
LOF	$15.8 \pm 0.0\,(15.8)$	$\mathbf{4.2} \pm 0.0\,(4.2)$	$5.2 \pm 0.0\,(5.2)$	$1.5 \pm 0.0\,(1.5)$	$0.0 \pm 0.0\,(0.0)$	TLE
IF	$12.9 \pm 2.6\,(19.8)$	$2.8 \pm 0.1\,(2.9)$	$10.1 \pm 4.5\,(22.6)$	$1.9 \pm 0.2\,(2.7)$	$0.1 \pm 0.0\,(0.1)$	$\mathbf{1.8} \pm 0.0\,(\underline{1.9})$
MLPAE	$52.8 \pm 9.9\,(64.5)$	$3.4 \pm 0.0\,(3.4)$	$5.9 \pm 0.8\,(7.9)$	$1.8 \pm 0.3\,(2.5)$	$0.1 \pm 0.0\,(0.1)$	$0.9 \pm 0.0\,(1.0)$
SCAN	$17.3 \pm 3.4\,(20.5)$	$3.3 \pm 0.0\,(3.3)$	$5.0 \pm 0.3\,(5.5)$	$2.0 \pm 0.1\,(2.1)$	$0.0 \pm 0.0\,(0.1)$	TLE
Radar	$\mathbf{92.1} \pm 0.7\,(\underline{92.9})$	$3.6 \pm 0.2\,(3.9)$	$7.2 \pm 0.0\,(7.2)$	$2.2 \pm 0.0\,(2.2)$	$\mathbf{0.2} \pm 0.0\,(0.2)$	OOM_C
ANOMALOUS	$\mathbf{92.1} \pm 0.7\,(\underline{92.9})$	$4.0 \pm 0.6\,(\underline{5.1})$	$7.2 \pm 0.0\,(7.2)$	$2.2 \pm 0.0\,(2.2)$	$\mathbf{0.2} \pm 0.0\,(0.2)$	OOM_C
GCNAE	$70.8 \pm 5.0\,(80.9)$	$3.4 \pm 0.0\,(3.4)$	$4.8 \pm 0.7\,(5.8)$	$2.1 \pm 0.4\,(3.5)$	$0.1 \pm 0.0\,(0.1)$	$1.0 \pm 0.0\,(1.0)$
DOMINANT	$18.0 \pm 10.2\,(36.2)$	$3.7 \pm 0.0\,(3.8)$	$7.6 \pm 5.0\,(23.2)$	$2.2 \pm 0.6\,(4.1)$	$0.1 \pm 0.1\,(\underline{0.4})$	OOM_C
DONE	$65.5 \pm 13.4\,(77.3)$	$3.7 \pm 0.4\,(4.5)$	$5.0 \pm 0.7\,(6.4)$	$1.8 \pm 0.3\,(2.6)$	$0.1 \pm 0.0\,(0.1)$	OOM_C
AdONE	$62.9 \pm 9.5\,(74.4)$	$3.3 \pm 0.4\,(4.0)$	$6.1 \pm 1.5\,(11.7)$	$2.5 \pm 0.3\,(3.2)$	$0.1 \pm 0.0\,(0.1)$	OOM_C
AnomalyDAE	$38.5 \pm 22.5\,(77.3)$	$3.7 \pm 0.1\,(3.8)$	$5.7 \pm 0.2\,(6.3)$	$\mathbf{3.5} \pm 1.4\,(\underline{7.8})$	$0.1 \pm 0.0\,(0.1)$	OOM_C
GAAN	$80.3 \pm 0.2\,(80.7)$	$3.7 \pm 0.1\,(3.9)$	$5.6 \pm 0.0\,(5.6)$	$2.6 \pm 0.8\,(5.6)$	$0.1 \pm 0.0\,(0.1)$	OOM_C
GUIDE	OOM_C	OOM_C	$4.8 \pm 0.9\,(6.9)$	$1.9 \pm 0.3\,(3.1)$	OOM_C	OOM_C
CONAD	$15.6 \pm 6.9\,(31.7)$	$3.7 \pm 0.3\,(4.6)$	$6.0 \pm 1.4\,(11.5)$	$2.5 \pm 0.8\,(4.9)$	$0.1 \pm 0.0\,(0.3)$	$0.9 \pm 0.0\,(0.9)$
Rec	$28.6 \pm 9.7\,(42.4)$	$2.9 \pm 0.1\,(3.2)$	$13.9 \pm 6.5\,(33.6)$	$2.9 \pm 0.8\,(6.4)$	$0.1 \pm 0.0\,(0.1)$	TLE
Rec (unweighted)	$29.2 \pm 10.5\,(43.0)$	$2.9 \pm 0.1\,(3.2)$	$14.6 \pm 7.0\,(30.8)$	$2.8 \pm 0.6\,(4.4)$	$0.1 \pm 0.0\,(0.2)$	TLE
Energy	$10.7 \pm 2.4\,(14.1)$	$4.0 \pm 0.4\,(5.0)$	$15.9 \pm 7.4\,(29.2)$	$2.7 \pm 0.7\,(4.4)$	$0.0 \pm 0.0\,(0.1)$	TLE
Energy (unweighted)	$11.0 \pm 2.9\,(18.0)$	$3.9 \pm 0.3\,(5.0)$	$\mathbf{17.2} \pm 8.0\,(29.0)$	$2.6 \pm 0.6\,(3.8)$	$0.0 \pm 0.0\,(0.0)$	TLE

Table 6. Recall@k (%) on datasets with organic outliers. The best average results are written in **bold**, and the best maximum results are underlined. TLE and OOM_C indicate that the method exceeded the time limit of 24 hours and failed to fit in VRAM, respectively.

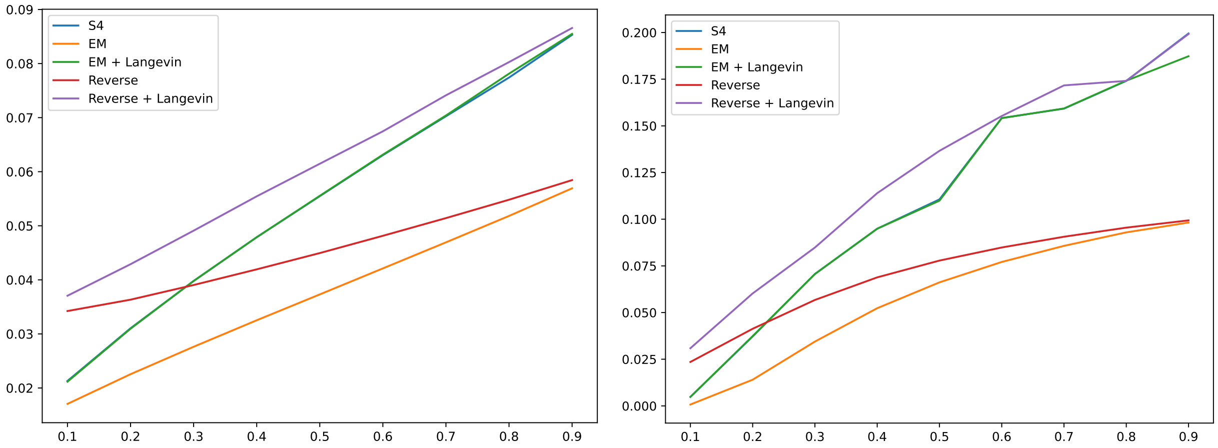
Algorithm	**Weibo**	**Reddit**	**Disney**	**Books**	**Enron**	**DGraph**
LOF	$22.0 \pm 0.0\,(22.0)$	$\mathbf{4.4} \pm 0.0\,(4.4)$	$0.0 \pm 0.0\,(0.0)$	$0.0 \pm 0.0\,(0.0)$	$0.0 \pm 0.0\,(0.0)$	TLE
IF	$13.8 \pm 6.4\,(24.3)$	$0.1 \pm 0.1\,(0.3)$	$9.2 \pm 8.3\,(16.7)$	$1.1 \pm 1.6\,(3.6)$	$0.0 \pm 0.0\,(0.0)$	$0.1 \pm 0.1\,(0.4)$
MLPAE	$48.9 \pm 11.0\,(62.1)$	$3.0 \pm 0.0\,(3.0)$	$0.0 \pm 0.0\,(0.0)$	$0.9 \pm 1.6\,(3.6)$	$0.0 \pm 0.0\,(0.0)$	$\mathbf{0.5} \pm 0.1\,(\underline{0.6})$
SCAN	$23.8 \pm 7.0\,(30.5)$	$2.7 \pm 0.3\,(3.0)$	$7.5 \pm 11.2\,(\underline{33.3})$	$0.7 \pm 1.4\,(3.6)$	$0.0 \pm 0.0\,(0.0)$	TLE
Radar	$\mathbf{86.4} \pm 0.8\,(\underline{87.4})$	$2.1 \pm 0.8\,(3.5)$	$0.0 \pm 0.0\,(0.0)$	$0.0 \pm 0.0\,(0.0)$	$0.0 \pm 0.0\,(0.0)$	OOM_C
ANOMALOUS	$\mathbf{86.4} \pm 0.8\,(\underline{87.4})$	$4.0 \pm 1.9\,(\underline{7.9})$	$0.0 \pm 0.0\,(0.0)$	$0.0 \pm 0.0\,(0.0)$	$0.0 \pm 0.0\,(0.0)$	OOM_C
GCNAE	$67.6 \pm 5.2\,(77.3)$	$3.0 \pm 0.0\,(3.0)$	$0.0 \pm 0.0\,(0.0)$	$0.7 \pm 1.8\,(7.1)$	$0.0 \pm 0.0\,(0.0)$	$0.4 \pm 0.0\,(0.4)$
DOMINANT	$19.7 \pm 13.8\,(37.4)$	$0.9 \pm 0.4\,(2.7)$	$3.3 \pm 6.7\,(16.7)$	$1.6 \pm 3.1\,(\underline{10.7})$	$0.0 \pm 0.0\,(0.0)$	OOM_C
DONE	$65.4 \pm 12.4\,(76.3)$	$2.8 \pm 1.6\,(5.7)$	$0.0 \pm 0.0\,(0.0)$	$1.1 \pm 1.6\,(3.6)$	$0.0 \pm 0.0\,(0.0)$	OOM_C
AdONE	$64.3 \pm 7.6\,(74.3)$	$1.0 \pm 1.2\,(3.8)$	$1.7 \pm 5.0\,(16.7)$	$\mathbf{3.0} \pm 1.7\,(7.1)$	$0.0 \pm 0.0\,(0.0)$	OOM_C
AnomalyDAE	$42.2 \pm 23.7\,(75.7)$	$0.9 \pm 0.5\,(3.0)$	$0.0 \pm 0.0\,(0.0)$	$2.7 \pm 2.2\,(7.1)$	$0.0 \pm 0.0\,(0.0)$	OOM_C
GAAN	$77.1 \pm 0.2\,(77.4)$	$1.1 \pm 0.4\,(2.2)$	$0.0 \pm 0.0\,(0.0)$	$1.8 \pm 1.8\,(3.6)$	$0.0 \pm 0.0\,(0.0)$	OOM_C
GUIDE	OOM_c	OOM_C	$0.0 \pm 0.0\,(0.0)$	$0.4 \pm 1.1\,(3.6)$	OOM_C	OOM_C
CONAD	$20.3 \pm 13.3\,(37.1)$	$1.3 \pm 1.6\,(7.6)$	$0.8 \pm 3.6\,(16.7)$	$1.7 \pm 2.9\,(\underline{10.7})$	$0.0 \pm 0.0\,(0.0)$	$0.4 \pm 0.1\,(\underline{0.6})$
Rec	$35.4 \pm 12.9\,(50.8)$	$2.2 \pm 0.8\,(4.4)$	$13.6 \pm 11.2\,(\underline{33.3})$	$2.1 \pm 2.6\,(7.1)$	$0.0 \pm 0.0\,(0.0)$	TLE
Rec (unweighted)	$36.4 \pm 14.5\,(52.7)$	$2.4 \pm 0.7\,(4.1)$	$14.2 \pm 12.5\,(\underline{33.3})$	$2.4 \pm 2.8\,(\underline{10.7})$	$0.0 \pm 0.0\,(0.0)$	TLE
Energy	$8.8 \pm 4.3\,(17.6)$	$4.2 \pm 1.4\,(7.1)$	$13.3 \pm 10.0\,(\underline{33.3})$	$2.7 \pm 3.2\,(\underline{10.7})$	$0.0 \pm 0.0\,(0.0)$	TLE
Energy (unweighted)	$9.2 \pm 5.0\,(24.2)$	$3.6 \pm 1.1\,(6.8)$	$\mathbf{15.8} \pm 12.3\,(\underline{33.3})$	$2.5 \pm 2.8\,(7.1)$	$0.0 \pm 0.0\,(0.0)$	TLE

E. Solver Comparison

For the reconstruction approach, we motivate our choice of solver (Euler-Maruyama) by evaluating the average error at different noise levels across the Books dataset. Formally, we compute each of the terms in Eq. (8) separately:

$$\mathrm{error}_X(\tau) = \frac{1}{|\mathcal{G}|}\sum_{v\in\mathcal{G}} \frac{\|\hat{\mathbf{X}}^{(j)}(v,\tau) - \mathbf{X}(v)\|_F}{N(v)\cdot F} \tag{15}$$

$$\mathrm{error}_A(\tau) = \frac{1}{|\mathcal{G}|}\sum_{v\in\mathcal{G}} \frac{\|\hat{\mathbf{A}}^{(j)}(v,\tau) - \mathbf{A}(v)\|_F}{(N(v))^2}. \tag{16}$$



(a) Feature matrix

(b) Adjacency matrix

Figure 1. The reconstruction errors of each graph component (y-axis) vs time τ (x-axis)

Figure 1 illustrates error_X and error_A for each solver at different reconstruction times τ. The reported errors are averaged across 20 runs with different hyperparameters. As can be seen from the plots, a simple Euler-Maruyama (EM) scheme consistently outperforms all the other solvers.

Both *EM* and *Reverse* (Song et al., 2020) correspond to different finite step discretization schemes of the underlying reverse-time SDE. The remaining methods (*S4* (Jo et al., 2022), *EM + Langevin*, *Reverse + Langevin*) are Predictor-Corrector solvers that leverage the Langevin MCMC as a corrector to improve the quality of intermediate samples (Song et al., 2020).

F. Energy Histograms

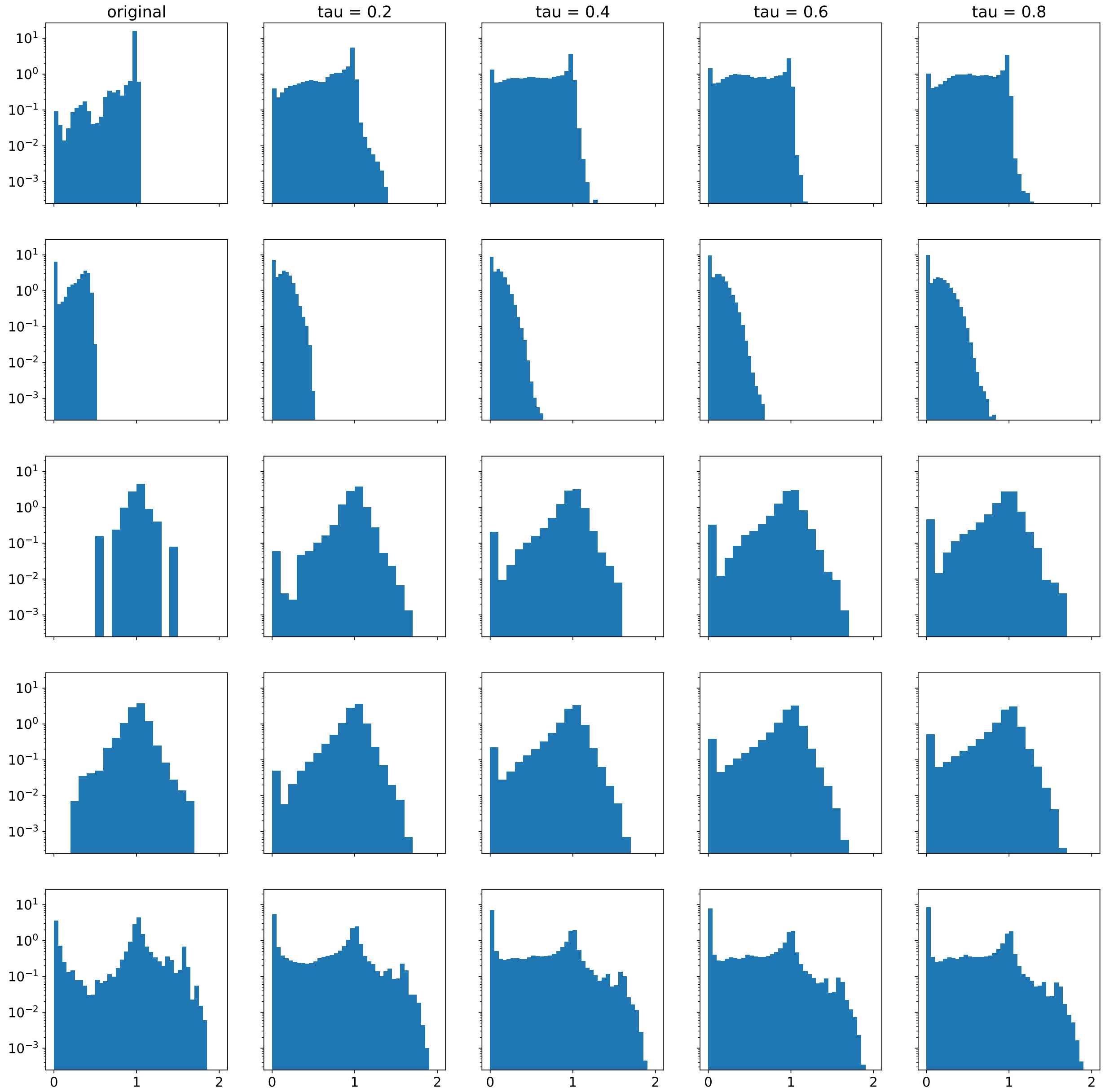


Figure 2. Histograms of the original and reconstructed normalized energies (log-densities). The datasets order from top to bottom: *Weibo*, *Reddit*, *Disney*, *Books*, *Enron*.

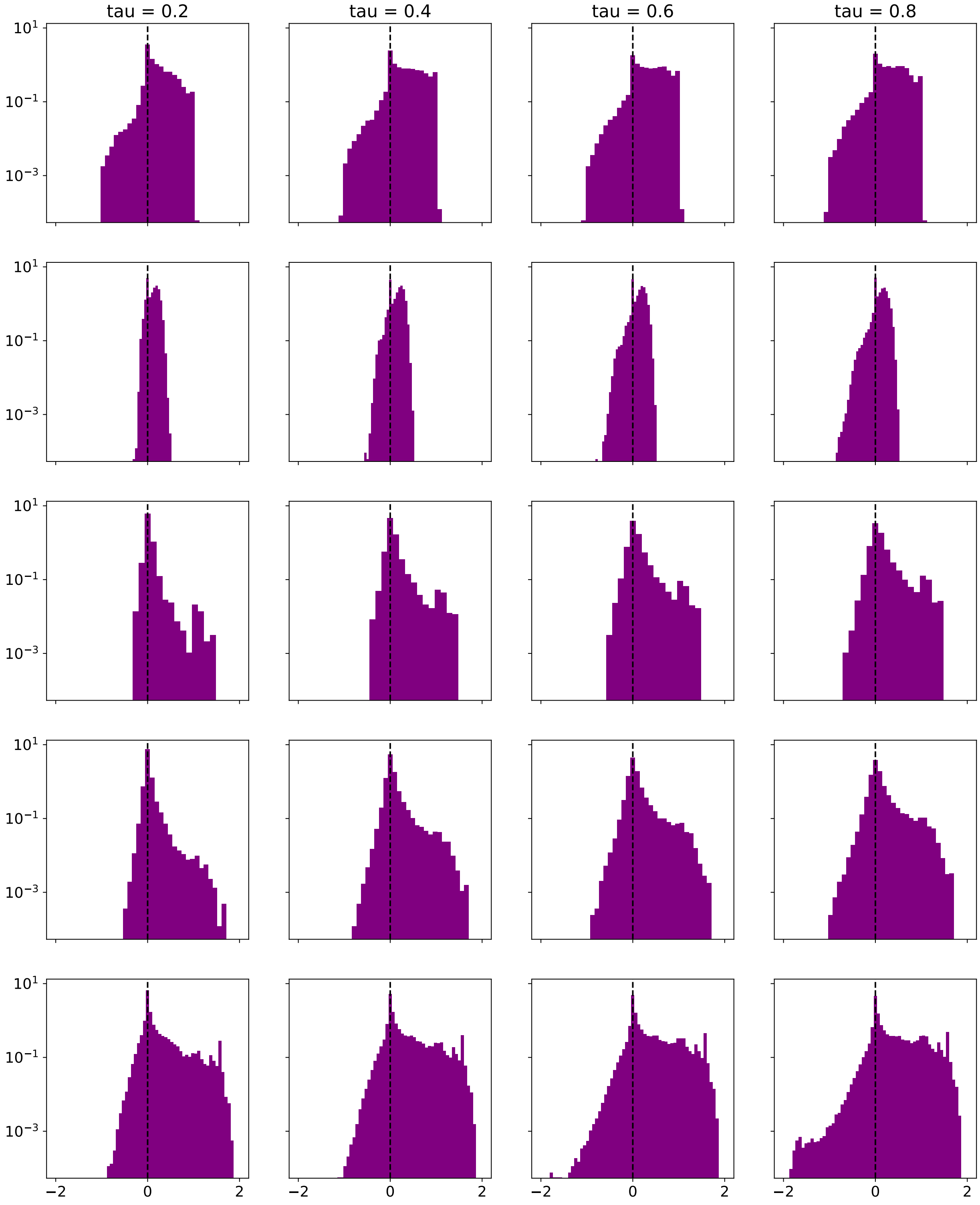
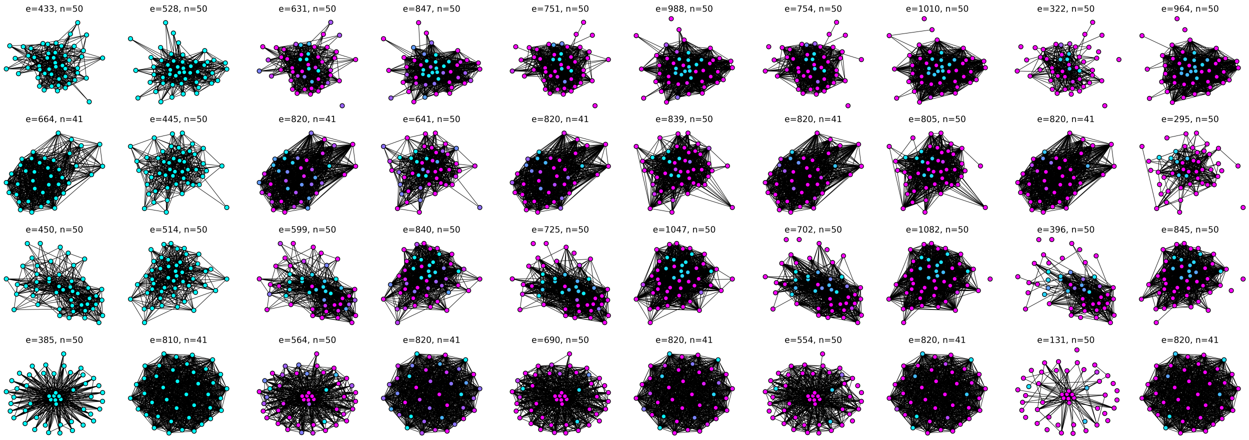


Figure 3. Histograms of the differences between the original and reconstructed normalized energies (log-densitites). Dashed black line is set to mark zero difference. The datasets order from top to bottom: *Weibo*, *Reddit*, *Disney*, *Books*, *Enron*.

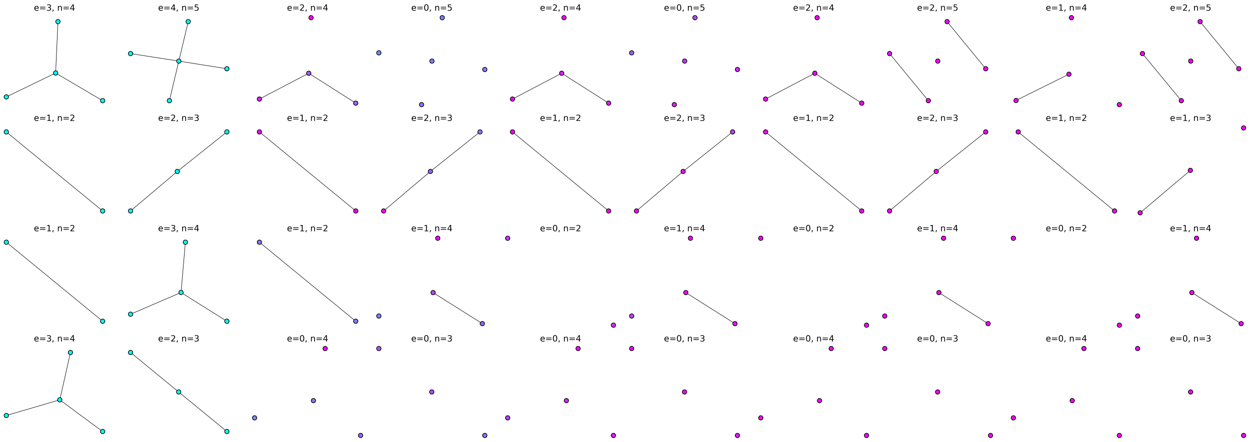
G. Ego-graphs

In this appendix, we plot 8 randomly selected ego-graphs and their reconstructions from each dataset. We reconstruct ego-graphs using checkpoints with index 0 (there are overall 20 checkpoints for each network). Graphs are summarized by the number of nodes and edges, denoted as n and e, respectively. Node color indicates the relative error in features, with cyan corresponding to perfect reconstruction, and magenta signifying that the error is comparable to the norm of features. The colors are interpolated using the *cool* colormap from Matplotlib (Hunter, 2007). The graph layout is computed using the Fruchterman-Reingold force-directed algorithm (Fruchterman & Reingold, 1991) (*spring_layout* in NetworkX package (Hagberg et al., 2008)).



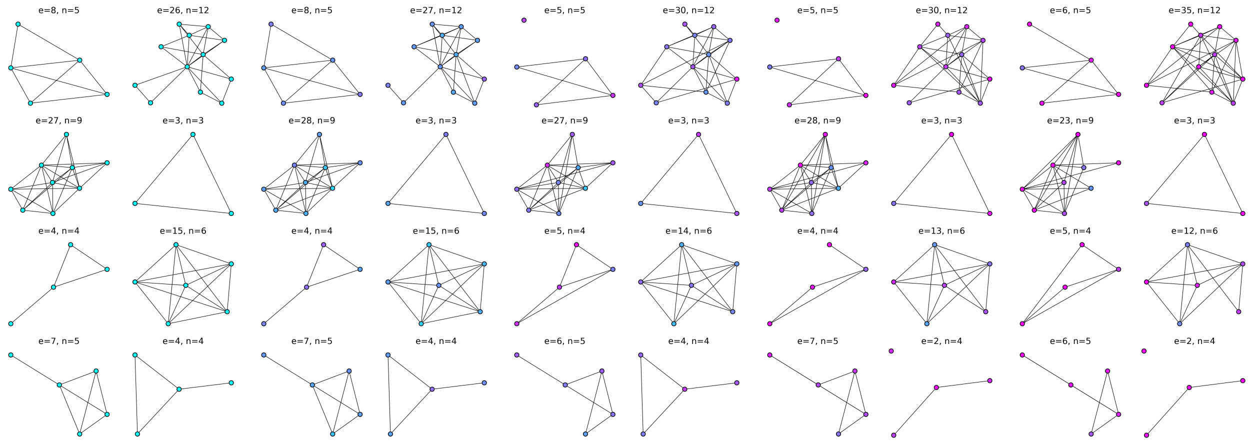
(a) Original (b) $\tau = 0.2$ (c) $\tau = 0.4$ (d) $\tau = 0.6$ (e) $\tau = 0.8$

Figure 4. The original and reconstructed ego-graphs from Weibo dataset.



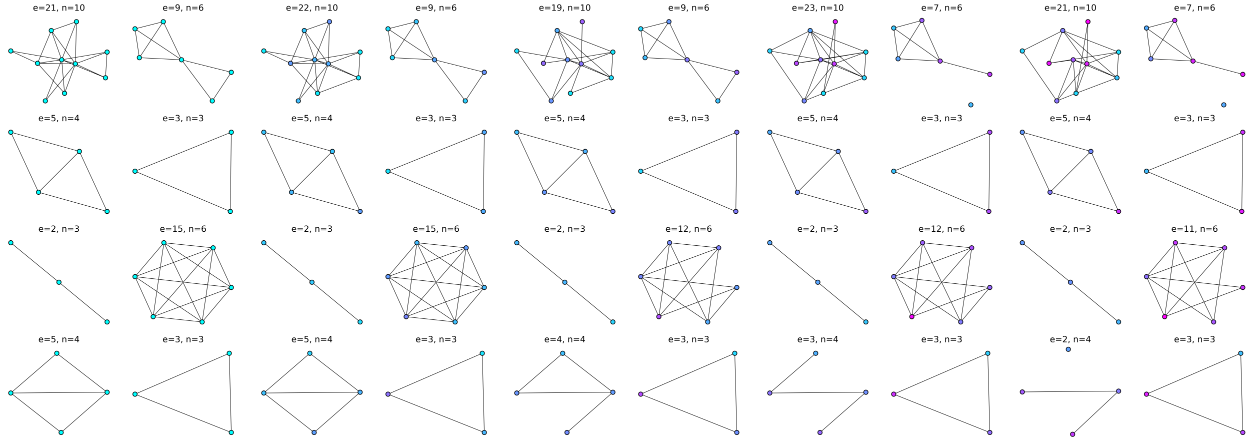
(a) Original (b) $\tau = 0.2$ (c) $\tau = 0.4$ (d) $\tau = 0.6$ (e) $\tau = 0.8$

Figure 5. The original and reconstructed ego-graphs from Reddit dataset.



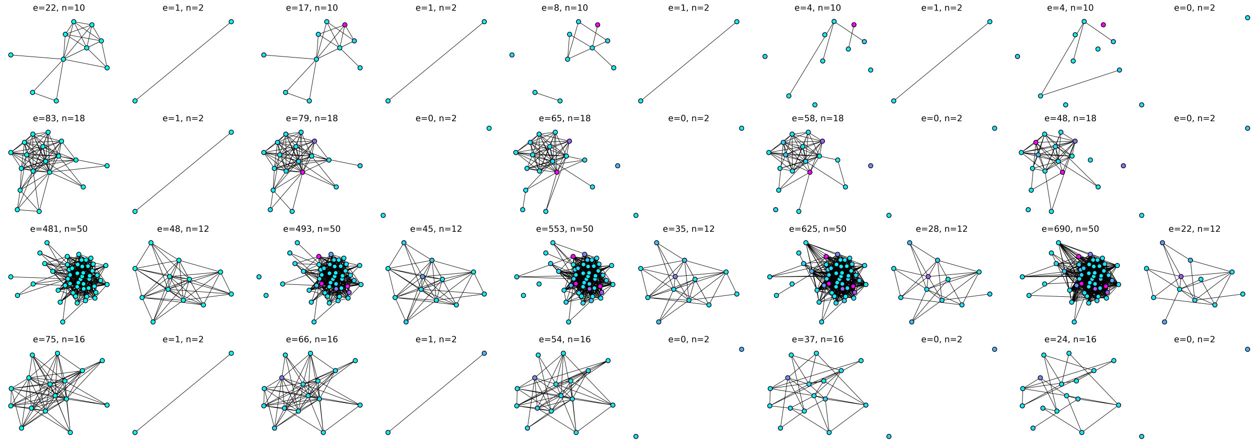
(a) Original (b) $\tau = 0.2$ (c) $\tau = 0.4$ (d) $\tau = 0.6$ (e) $\tau = 0.8$

Figure 6. The original and reconstructed ego-graphs from Disney dataset.



(a) Original (b) $\tau = 0.2$ (c) $\tau = 0.4$ (d) $\tau = 0.6$ (e) $\tau = 0.8$

Figure 7. The original and reconstructed ego-graphs from Books dataset.



(a) Original (b) $\tau = 0.2$ (c) $\tau = 0.4$ (d) $\tau = 0.6$ (e) $\tau = 0.8$

Figure 8. The original and reconstructed ego-graphs from Enron dataset.