Graph Diffusion Models for Anomaly Detection

Zekuan Liu, Huijun Yu, Yao Yan, Ziqing Hu, Pankaj Rajak, Amila Weerasinghe,
Olcay Boz, Deepayan Chakrabarti, Fei Wang
{lzekuan,huijuyu,ynyao,ziqinghu,rajakpan,weera,olcayboz,deepayc,fiwan}@amazon.com
Amazon
Seattle, Washington, USA

ABSTRACT

Anomaly detection on graphs focuses on identifying irregular patterns or anomalous nodes within graph-structured data, which deviate significantly from the norm. This domain gains paramount importance due to its wide applicability in various fields such as spam detection, anti-money laundering, and network security. In the application of anomaly detection on graphs, tackling the challenges posed by label imbalance and data insufficiency is of significance. Recent proliferation in generative models, especially diffusion models, paves a promising way. In this paper, we introduce a graph diffusion model in latent space, designed to alleviate the label imbalance problem prevalent in anomaly detection on graphs. The proposed model is capable of multitask generation of graph structures and node features, and further endowed with conditional generative capabilities to produce only positive examples, thereby mitigating label imbalance issues. We improved the diffusion model to apply on both homogeneous graphs and heterogeneous graphs. Through extensive experiments, we demonstrate that our proposed method offers notable improvements over conventional techniques.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; • **Mathematics of computing** → *Graph algorithms*; • **Computer systems organization** → *Neural networks.*

KEYWORDS

Anomaly Detection, Graph Neural Networks, Diffusion Models

ACM Reference Format:

Zekuan Liu, Huijun Yu, Yao Yan, Ziqing Hu, Pankaj Rajak, Amila Weerasinghe,, Olcay Boz, Deepayan Chakrabarti, Fei Wang. 2018. Graph Diffusion Models for Anomaly Detection. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Anomaly detection has always been a critical task in the realm of machine learning and data mining applications, due to its significance in identifying irregular patterns that deviate from the norm in large datasets. These anomalies often correspond to critical, actionable insights in various domains such as spam detection, network security, and health monitoring. The importance of accurate and efficient anomaly detection on graphs is further underscored in the web era, where interconnected data proliferates at an unprecedented rate. The interconnected nature of these data sources, often represented as graphs, amplifies the complexity of detecting anomalies. Graphs, which effectively capture relationships and structural information, are pivotal in understanding the interactions in diverse applications ranging from social networks to biological networks.

However, the task of anomaly detection in graphs presents unique challenges. The complexity of graph structures, which include nodes, edges, and their attributes, requires sophisticated algorithms capable of capturing both local and global irregularities. Traditional methods, which predominantly focus on point anomalies in tabular data, are insufficient for graph data due to its intrinsic relational information and high dimensionality. Furthermore, the dynamic nature of graphs, where nodes and edges can evolve over time, necessitates algorithms that can adapt and detect anomalies in a continuously changing environment.

Recent advancements in graph mining techniques, particularly in the context of deep learning, have opened new avenues for addressing these challenges. Graph neural networks (GNNs) learn complex patterns and dependencies within graph data, and have shown promising results in identifying anomalies on graphs. Yet, the application of these advanced techniques raises additional considerations, similar to anomaly detection on tabular data. Anomaly detection on graphs also faces the challenge of label imbalance, where the number of anomaly examples is significantly lower than benign examples. Label imbalance can result in the downgraded performance of anomaly detection.

The advent of diffusion models in the field of generative modeling marks a significant milestone, particularly in synthesizing high-quality data representations. Originating from the domain of image generation, these models have demonstrated an unprecedented ability to capture intricate data distributions. In essence, a diffusion model operates through a gradual process of noising and denoising data, where the original data distribution is incrementally corrupted with noise, followed by a learned reverse process that denoises this data back to its original distribution. This iterative approach enables the model to capture complex, high-dimensional data distributions effectively. While their application has been predominantly in image synthesis, the potential of diffusion models in other domains, such as anomaly detection, especially on graph data remain unexplored.

To bridge this gap, this study proposes a novel diffusion model based graph generator. Our generator use graph autoencoder to encode graph and applies diffusion model in the latent space. The

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

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

generator generates graph structure and node feature simultaneously in a multitask fashion. It offers a unique capability to generate positive (anomaly) examples, thus effectively alleviating the problem of label imbalance. Also, it is able to generate heterogeneous graphs with heterogeneous generator. Adding the generated positive examples to the training set, it provides a relatively balanced dataset for training downstream anomaly detection, and alleviates the label imbalance problem in anomaly detection on graphs.

The contribution in this project can be mainly summarized in to following two parts:

- We propose a diffuision model based graph generator that capable to generate heterogeneous graph structure and node features simultaneously conditioned on positive label.
- We further alleviate the label imbalance problem in general anomaly detection on graphs framework.
- Experimental results on multiple datasets prove that our generator outperforms baselines. Ablation study demonstrated the effectiveness of each components of our generator.

The remaining paper is organized as follows: after this introduction, we delineate previous research followed by the methodology, featuring the architecture of our diffusion model based graph generator and its components. This is followed by rigorous empirical evaluations, including ablation studies to assess the effectiveness of each component and case studies for an in-depth understanding of generated features. Finally, we summarize our contributions and explore avenues for future research.

2 RELATED WORKS

Anomaly Detection on Graphs has emerged as a critical area of research, distinguished by unique challenges associated with the complex nature of graph structures. Early methods focused on clustering and proximity-based techniques, which were adept at handling anomalies in simpler graph but faltered when applied to the intricate relationships inherent in complex graph data [15]. The introduction of GNNs marked a significant shift, offering a more nuanced approach to modeling relational data. GNNs, through their capacity to encapsulate both node-level and graph-wide patterns, have demonstrated considerable success in detecting anomalies across a range of applications, from fake news detection to anti-money laundering [1, 8, 9, 14]. However, label imbalance challenge persists, particularly in adapting these models to heterogeneous graphs [17] where the graph contains different types of nodes and edges. The problem is more complicated if the graph is dynamic.

Graph Generators have gained attention as a potent tool for addressing the data scarcity and synthetic data generation challenges in graph-based applications. Conventional methods adopt autoregressive paradigm to generate graphs [16]. Recently. the development of generative models, particularly diffusion models, has been a noteworthy advancement. Originating in the domain of image synthesis, these models have demonstrated a unique capability in capturing complex data distributions through iterative noising and denoising processes [3]. Their application in graph data is relatively new but promising, offering a method to generate realistic graph structures and node features that mirror the intricacies of real-world graphs [13]. This capability is particularly valuable in exploring the structural variability of graphs and understanding how anomalies manifest in different graph contexts.

Label Imbalance in anomaly detection presents a significant hurdle, particularly in graph data where anomalies are naturally rare and often overshadowed by the majority of normal instances. This imbalance leads to models that are biased towards the majority class, thereby diminishing their effectiveness in identifying true anomalies. Some works adopt interpolation based methods to address label imbalance [18], and some others use data augmentations [7]. The integration of graph generators offers a novel solution to this problem. By generating synthetic anomalies, these models can improve datasets, creating a more balanced landscape for model training and evaluation.

3 METHODOLOGY

To alleviate the label imbalance problem in our anomaly detection, this study proposes a diffusion model based graph generator to generate synthetic anomaly nodes within a heterogeneous graph structure. The overview of the proposed generator is shown in Figure 1. During the training Phase, the generator is trained solely on the real training data. Then the generator can generate synthetic graphs with fraudulent nodes. In downstream application, these synthetic data, in conjunction with real training data, are utilized for training the anomaly detection model. This section includes the problem definition and how we generate the graph. In addition, the details of the diffusion model will be introduced.

3.1 Problem Definition

Our primary objective is to enrich a dataset with anomaly nodes for robust training of anomaly detection systems. These synthetic nodes encompass four components:

- **Node Feature**: Attributes associated with the node within the graph.
- **Node Timestamp**: A temporal stamp related to the node.
- **Positive Label**: The positive label denoting the anomaly nature of the node.
- **Heterogeneous Graph Structure**: Different type of nodes and edge connections.

Each of these components is generated through a sequence of systematic steps, elaborated in the subsequent subsections.

3.2 Encoding with Graph Autoencoder

In order to encode our anomaly detection graphs into a latent space for the diffusion model, we adopt a graph autoencoder. For graph data, which inherently possesses a complex and multifaceted nature, including node attributes and graph topology, a specialized design is imperative. This complexity arises from the interconnectedness and the structural dependencies within the graph, which are not present in traditional tabular data. Therefore, the design of the graph autoencoder needs to address these unique characteristics.

3.2.1 Generating Node Attributes. We first apply a traditional autoencoder to cope with the node attributes generation. An autoencoder is a type of artificial neural network used to learn efficient representations (encodings) of data, typically for the purpose of

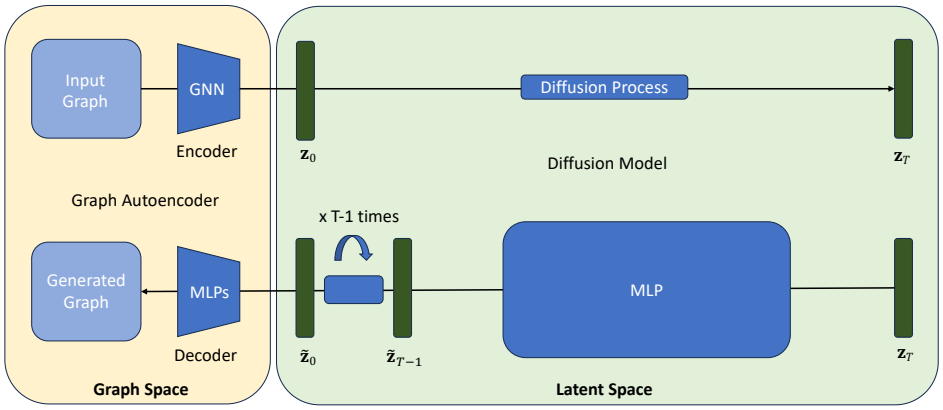

Figure 1: An overview of proposed diffusion model based graph generator. We first encoder the input graph to latent space with a GNN encoder. The diffusion process gradually adds noise to the embedding by T steps, while the MLP denoises the embedding T times. Finally, the decoder reconstructs the graph from the latent embedding.

dimensionality reduction. Its architecture is characterized by a two-part structure: the encoder and the decoder. The encoder of an autoencoder is responsible for transforming the input data into a lower-dimensional latent space. This process involves a series of layers that gradually compress the input data, extracting and retaining the most salient features. The encoder's primary objective is to learn a compressed latent representation of the input data that encapsulates its essential characteristics, thereby reducing its dimensionality while preserving its significant attributes.

Following the encoding process, the decoder reconstructs the input data from the condensed representation in the latent space. The goal of the decoder is to generate an output that closely approximates the original input data, using the compressed information encoded by the encoder. This reconstruction process is crucial, as it ensures that the learned representations in the latent space are meaningful and informative, capturing the intrinsic patterns and structures of the input data.

The training of an autoencoder involves adjusting the weights of the network to minimize the difference between the original input and its reconstruction, typically using a loss function like mean squared error (MSE). Through this process, the autoencoder learns to prioritize the most significant features in the input data, effectively learning a compressed but informative representation.

As an enhanced variant of traditional autoencoder, the Variational Autoencoder (VAE) introduces a probabilistic approach, increasing the generalizing ability. Specifically, In a VAE, the encoder predicts two parameters for the prior Gaussian distribution—mean (μ) and standard deviation (σ). During the forward pass, a sample is drawn from the distribution defined by these parameters, which is then decoded to reconstruct the input data. The VAE loss function is a sum of two terms: reconstruction term, which is a MSE between the input and the reconstructed output, and a regularization term, which is KL divergence between the learned distribution and a standard Gaussian distribution.

3.2.2 Conditional Generation for Only Nodes with Positive Label. In our approach, we aim to enhance the control over the generation of node types, particularly differentiating between anomalous and normal nodes. To achieve this, we have incorporated conditional variables into the VAE. It enables the model to factor in the additional label information during the generation process.

During the training phase of the VAE, we concatenate the labels indicating the node type (anomalous or normal) with their respective feature vectors. This concatenated data is then fed into the encoder of the VAE. The encoder, thus, learns a latent representation that encapsulates not only the features of the nodes but also their corresponding labels.

In the generation phase, the decoder of the VAE is explicitly conditioned on these labels. This conditioning is pivotal as it directs the decoder to generate feature vectors that are inherently aligned with the specified labels. Consequently, when the label indicates 'fraudulent', the decoder is steered to produce feature vectors that are characteristic of fraudulent nodes.

This methodology allows for a more controlled generation of nodes, enhancing the model's ability to differentiate and generate distinct types of nodes based on their underlying characteristics. Such an approach is particularly beneficial in scenarios like fraud detection, where the distinction between normal and anomalous behavior is crucial for effective model performance.

3.2.3 Generating Heterogeneous Graph Structure. In our methodology for generating the graph structure, the Variational Graph Autoencoder (VGAE) [6] is utilized. This approach adapts the architecture of a VAE to graph-based data, enabling the encoder to effectively capture both topological and feature information of the graph into a latent representation. In turn, the decoder focuses on reconstructing the graph structure. Our model is specifically tailored for multi-task learning, leveraging a shared GNN as the encoder. This encoder is central to our approach, as it processes the graph structure and node features simultaneously, embedding them into a latent space.

The decoder is bifurcated into two separate entities, each with a distinct role. The first decoder is dedicated to feature reconstruction, while the second focuses on graph structure. This bifurcation is pivotal in addressing the distinct aspects of our graph data, namely, the node features and the graph topology. The VGAE model is

inherently aligned with the framework of a traditional VAE, where the loss function comprises two components: the reconstruction loss and the Kullback-Leibler (KL) divergence. These components are critical as they collectively cater to both feature and graph structure generation tasks within our model.

However, our research addresses a graph that is inherently heterogeneous, encompassing multiple types of nodes and edges. This complexity necessitates a modification in the VGAE framework. Consequently, we substitute the standard VGAE encoder with a Heterogeneous Graph Transformer (HGT), as introduced by [4]. The HGT is adept at handling the diverse and complex nature of our graph, allowing for a more nuanced understanding and processing of the heterogeneous elements.

Additionally, we implement separate decoders for different types of edges. This differentiation is essential for accurately generating adjacency matrices specific to each edge type in our heterogeneous graph. The use of distinct decoders enables us to tailor the reconstruction process for each edge type, thereby enhancing the fidelity of the reconstructed heterogeneous graph.

3.2.4 Timestamp Generation. In the proposed methodology, a distinct temporal generator is meticulously designed and deployed, specifically tasked with the generation of timestamps. This component parallels the architecture of the feature generator in terms of its foundational structure; however, it diverges in its targeted output, which is singularly dimensional. The primary output of this temporal generator is a continuous variable representing the time aspect of the data, encapsulating a critical dimension in temporal data analysis.

To ensure the precision and reliability of the temporal generator, a Mean Squared Error (MSE) loss function is employed. This choice of loss function is particularly effective for regression tasks, where the goal is to minimize the average squared difference between the estimated values and the actual value. In the context of this research, the MSE loss function aids in fine-tuning the temporal generator to produce timestamps that closely align with the true temporal characteristics of the dataset.

3.3 Diffusion Model in Latent Space

With the graph autoencoder described above, we are able to generate the synthetic anomalies. However, when due to the complexity of the graph structure, the divergence between prior distribution and data distribution can be large, thereby result in suboptimal performance in the downstream anomaly detection. As such, we propose to further apply a diffusion model in latent space for better generation quality.

In this paper, we adopt DDPM [3]. It is mathematically formulated as:

$$p_\theta(\mathbf{z}_0) = \int p_\theta(\mathbf{z}_{0:T})\, d\mathbf{z}_{1:T}, \tag{1}$$

where $\mathbf{z}_1, \ldots, \mathbf{z}_T$ represent latent variables of the same dimensionality as the data $\mathbf{z}_0 \sim q(\mathbf{z}_0)$. The joint distribution $p_\theta(\mathbf{z}_{0:T})$ is known as the reverse process. This process is a Markov chain initiated from a standard Gaussian distribution $p(\mathbf{z}_T) = \mathcal{N}(\mathbf{z}_T; \mathbf{0}, \mathbf{I})$ and evolves with learned Gaussian transitions, defined as:

$$p_\theta(\mathbf{z}_{0:T}) = p(\mathbf{z}_T) \prod_{t=1}^{T} p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t), \tag{2}$$

$$p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{z}_t, t), \Sigma_\theta(\mathbf{z}_t, t)). \tag{3}$$

A distinct characteristic of diffusion models is the forward diffusion process, an approximate posterior $q(\mathbf{z}_{1:T}|\mathbf{z}_0)$. This process is a fixed Markov chain that incrementally introduces Gaussian noise into the data based on a variance schedule $\beta_1, \ldots, \beta_T$:

$$q(\mathbf{z}_{1:T}|\mathbf{z}_0) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}), \tag{4}$$

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t \mathbf{I}). \tag{5}$$

The training of diffusion models involves optimizing the variational bound on the negative log likelihood:

$$L = \mathbb{E}[-\log p_\theta(\mathbf{z}_0)] \leq \mathbb{E}_q \left[-\log p(\mathbf{z}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)}{q(\mathbf{z}_t|\mathbf{z}_{t-1})} \right]. \tag{6}$$

The variance parameters β_t of the forward process can either be learned through reparameterization or set as fixed hyperparameters. The reverse process is made expressive, particularly by the choice of Gaussian conditionals in $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$, especially when the β_t values are small. A notable aspect of the forward process is the ability to sample $\mathbf{z}_t$ at any timestep t in a closed form. This is facilitated by the definitions $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The sampling is:

$$q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t; \sqrt{\bar{\alpha}_t}\mathbf{z}_0, (1-\bar{\alpha}_t)\mathbf{I}). \tag{7}$$

Efficient training is feasible by optimizing random terms of L using stochastic gradient descent. Additional improvements in training are achieved through variance reduction. This is done by reformulating L (from Eq. 6) as:

$$\mathbb{E}_q \{ \underbrace{\mathrm{KL}[q(\mathbf{z}_T|\mathbf{z}_0)||p(\mathbf{z}_T)]}_{L_T} + \sum_{t>1} \underbrace{\mathrm{KL}[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0)||p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)]}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{z}_0|\mathbf{z}_1)}_{L_0} \}. \tag{8}$$

Equation 8 uses the Kullback-Leibler (KL) divergence to compare $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ against the tractable forward process posteriors conditioned on $\mathbf{z}_0$. The expressions for the posteriors are:

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0) = \mathcal{N}(\mathbf{z}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{z}_t, \mathbf{z}_0), \tilde{\beta}_t \mathbf{I}), \tag{9}$$

where $\tilde{\boldsymbol{\mu}}_t(\mathbf{z}_t, \mathbf{z}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{z}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{z}_t$ and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

Since all KL divergences in Equation 8 compare Gaussians, they can be computed in a Rao-Blackwellized manner using closed-form expressions instead of Monte Carlo estimates with high variance.

4 EXPERIMENTS

4.1 Experimental Setup

To comprehensively evaluate the performance of our proposed graph generator, we compared it against baseline methods, including simple reweighting and only graph autoencoder (GAE) without the diffusion model. The backbones of the baselines contain on the

Table 1: Anomaly Detection Performance Comparison Over Different Datasets.

Metrics	Dataset	GCN	GraphSAGE	GAT	GT	GAE	Diffusion
AUROC	**reddit**	0.6085	0.6550	0.6671	0.6436	0.6719	**0.6735**
	weibo	0.9820	0.9655	0.9406	0.9651	0.9841	**0.9882**
	amazon	0.8237	0.8802	**0.9778**	0.9086	0.9624	0.9745
	tfinance	0.9361	0.8042	0.9412	0.8409	0.9471	**0.9700**
	dgraph	0.7562	0.7559	0.7603	0.7582	0.7594	**0.7604**
AUPRC	**reddit**	0.0439	0.0622	0.0641	0.0593	0.0681	**0.0698**
	weibo	0.9344	0.8926	0.9025	0.9042	0.9479	**0.9489**
	amazon	0.3347	0.6061	0.8846	0.7656	0.8584	**0.8922**
	tfinance	0.7652	0.1468	0.6745	0.2676	0.6968	**0.8874**
	dgraph	0.0385	0.0375	**0.0390**	0.0386	**0.0390**	**0.0390**
Recall@K	**reddit**	0.0476	0.0884	0.0612	0.1020	**0.1088**	**0.1088**
	weibo	0.8963	0.8646	0.8674	0.8530	0.9107	**0.9222**
	amazon	0.3750	0.6413	0.8424	0.7554	0.8315	**0.8587**
	tfinance	0.7143	0.1678	0.6976	0.3204	0.7074	**0.8294**
	dgraph	0.0679	0.0735	0.0752	**0.0774**	0.0765	0.0739

top of various GNNs, including GCN [5], GraphSAGE [2], GAT [12], and Graph Transformer (GT) [10]. We opted for three metrics that are robust to label imbalance: Area Under the Receiver Operating Characteristic Curve (AUROC), Area Under the Precision Recall Curve (AUPRC), and Recall@k, where k is the number of anomalies in the ground truth label.

4.2 Datasets

We follow previous work [11], and conducted experiments on five distinct datasets, characterized by varying scales and label imbalances. The statistics of the dataset are presented in Table 2.

Table 2: The statistics of the datasets.

	Nodes	Edges	Attr.	Ratio	Etypes	Time
reddit	10,984	168,016	64	3.30%	1	×
weibo	8,405	407,963	400	10.30%	1	×
tfinance	39,357	21,222,543	10	4.60%	1	×
amazon	11,944	4,398,392	25	9.50%	3	×
dgraph	3,700,550	4,300,999	17	1.30%	11	✓

4.3 Experimental Results

In the experimental evaluation, as delineated in Table 1, our proposed method demonstrates superior performance compared to the established baselines across a majority of the test scenarios. Particularly noteworthy is the performance on the tfinance dataset, where our generator model achieves a significant improvement in AUROC, despite the high performance on baselines. The AUROC value escalates to 0.97, up from 0.94, when compared to GAT, underscoring the efficacy of our approach in this context.

Further analysis reveals the integral role of the diffusion model in our framework. By conducting a comparative study with GAE, which notably omits the diffusion component but keep the variational autoencoder parts , we observe a marked enhancement in performance with the inclusion of our diffusion model. This comparison not only highlights the quantitative improvements afforded by our method but also qualitatively underscores the necessity of the diffusion model in our graph generator.

4.4 Case Study

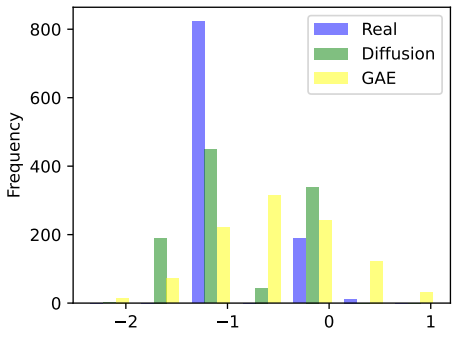

Figure 2: Distribution comparison of one dimension node feature between real data and generated data in DGraph dataset.

In Figure 2, we visualize the data generated by GAE and our generator compared with the real data. From the figure, we can observe that the data generated by our diffusion model based generator is closer to real data in distribution, while GAE can only generate a Gaussian distribution.

5 CONCLUSION

In this research, we propose a novel diffusion model based graph generator operating within a latent space framework, specifically engineered to address the challenges posed by label imbalance in graph-based anomaly detection. This model demonstrates proficiency in the simultaneous generation of graph structures and node features, incorporating multitasking capabilities. Furthermore, it is uniquely equipped with conditional generative functions, enabling the selective generation of positive examples. It is instrumental in effectively counteracting the prevalent issue of label imbalance.

REFERENCES

[1] Yingtong Dou, Kai Shu, Congying Xia, Philip S Yu, and Lichao Sun. 2021. User preference-aware fake news detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2051–2055.

[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[4] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*. 2704–2710.

[5] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[6] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[7] Fanzhen Liu, Xiaoxiao Ma, Jia Wu, Jian Yang, Shan Xue, Amin Beheshti, Chuan Zhou, Hao Peng, Quan Z Sheng, and Charu C Aggarwal. 2022. Dagad: Data augmentation for graph anomaly detection. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 259–268.

[8] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. 2022. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems* 35 (2022), 27021–27035.

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

[10] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509* (2020).

[11] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. 2023. GADBench: Revisiting and Benchmarking Supervised Graph Anomaly Detection. *arXiv preprint arXiv:2306.12251* (2023).

[12] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

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

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

[15] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. 2007. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 824–833.

[16] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*. PMLR, 5708–5717.

[17] Jianan Zhao, Xiao Wang, Chuan Shi, Zekuan Liu, and Yanfang Ye. 2020. Network schema preserving heterogeneous information network embedding. In *International joint conference on artificial intelligence (IJCAI)*.

[18] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*. 833–841.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009