Graph Fairing Convolutional Networks for Anomaly Detection

Mahsa Mesgaran and A. Ben Hamza
Concordia Institute for Information Systems Engineering
Concordia University, Montreal, QC, Canada

Abstract

Graph convolution is a fundamental building block for many deep neural networks on graph-structured data. In this paper, we introduce a simple, yet very effective graph convolutional network with skip connections for semi-supervised anomaly detection. The proposed layerwise propagation rule of our model is theoretically motivated by the concept of implicit fairing in geometry processing, and comprises a graph convolution module for aggregating information from immediate node neighbors and a skip connection module for combining layer-wise neighborhood representations. This propagation rule is derived from the iterative solution of the implicit fairing equation via the Jacobi method. In addition to capturing information from distant graph nodes through skip connections between the network's layers, our approach exploits both the graph structure and node features for learning discriminative node representations. These skip connections are integrated by design in our proposed network architecture. The effectiveness of our model is demonstrated through extensive experiments on five benchmark datasets, achieving better or comparable anomaly detection results against strong baseline methods. We also demonstrate through an ablation study that skip connection helps improve the model performance.

Keywords: Anomaly detection; graph convolutional network; skip connection; implicit fairing; Jacobi method.

1 Introduction

Anomaly detection is of paramount importance when deploying artificial intelligence systems, which often encounter unexpected or abnormal items/events that deviate significantly from the majority of data. Anomaly detection techniques are widely used in a variety of real-world applications, including, but not limited to, intrusion detection, fraud detection, healthcare, Internet of Things, Industry 4.0 and beyond, surveillance, and social networks [1–3]. Most of these techniques often involve a training set where no anomalous instances are encountered, and the challenge is to identify suspicious items or events, even in the absence of abnormal instances. In practical settings, the output of an anomaly detection model is usually an alert that triggers every time there is a anomaly or a pattern in the data that is atypical.

Detecting anomalies is a challenging task, primarily because the anomalous instances are not known a priori and also the

vast majority of observations in the training set are normal instances. Therefore, the mainstream approach in anomaly detection has been to separate the normal instances from the anomalous ones by using unsupervised learning models. One-class support vector machines (OC-SVM) are a classic example [4], which is a one-class classification model trained on data that has only one class (i.e. normal class) by learning a discriminative hyperplane boundary around the normal instances. Another commonly-used anomaly detection approach is support vector data description (SVDD) [5], which basically finds the smallest possible hypersphere that contains all instances, allowing some instances to be excluded as anomalies. Zhang *et al.* [6] presented a graph model-based multiscale feature fitting method for unsupervised anomaly detection and localization. Arias *et al.* [7] introduced an unsupervised parameter-free analytic isolation and distance-based anomaly detection algorithm, which integrates both distance and isolation metrics. However, these approaches rely on hand-crafted features, are unable to appropriately handle high-dimensional data, and often suffer from computational scalability issues.

Deep learning has recently emerged as a very powerful way to hierarchically learn abstract patterns from data, and has been successfully applied to anomaly detection, showing promising results in comparison with shallow methods [8]. Ruff *et al.* [9] extend the shallow one-class classification SVDD approach to the deep learning setting by proposing a deep learning based SVDD framework for anomaly detection using an anomaly detection based objective. Deep SVDD is an unsupervised learning model that learns to extract the common factors of variation of the data distribution by training a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data. Also, Ruff *et al.* [10] introduce a deep semi-supervised anomaly detection (Deep SAD) approach, which is a generalization of the unsupervised Deep SVDD technique to the semi-supervised setting. Deep SAD differs from Deep SVDD in that its objective function also includes a loss term for labeled normal and anomalous instances. The more diverse the labeled anomalous instances in the training set, the better the anomaly detection performance of Deep SAD. The key difference between deep one-class models such as Deep SVVD and semi-supervised anomaly detection methods such as Deep SAD lies in the way they are trained and the amount of labeled data required. The former is an unsupervised anomaly detection method that requires only normal data during training. It is trained to learn a representation of normal data, which is then used to distinguish between normal and anomalous data. Semi-supervised anomaly detection, on the other hand, is a hybrid

approach that generally uses both normal and anomalous data during training. The model is trained using both labeled and unlabeled data, where the labeled data consists of a small portion of anomalous data and a large portion of normal data. The main advantage of semi-supervised anomaly detection is that it can achieve higher accuracy than unsupervised methods, as the use of labeled data during training provides additional information, enabling the model to better distinguish between normal and anomalous data points.

Owing to the recent developments in deep semi-supervised learning on graph-structured data, there has been a surge of interest in the adoption of graph neural networks for learning latent representations of graphs [11, 12]. Defferrard *et al.* [11] introduce the Chebyshev network, an efficient spectral-domain graph convolutional neural network that uses recursive Chebyshev polynomial spectral filters to avoid explicit computation of the Laplacian spectrum. These filters are localized in space, and the learned weights can be shared across different locations in a graph. An efficient variant of graph neural networks is graph convolutional networks (GCNs) [12], which is an upsurging semi-supervised graph-based deep learning framework that uses an efficient layer-wise propagation rule based on a first-order approximation of spectral graph convolutions. The feature vector of each graph node in GCN is updated by essentially applying a weighted sum of the features of its immediate neighboring nodes. While significant strides have been made in addressing anomaly detection on graph-structured data [13], it still remains a daunting task on graphs due to various challenges, including graph sparsity, data nonlinearity, and complex modality interactions [14]. Ding *et al.* [14] design a GCN-based autoencoder for anomaly detection on attributed networks by taking into account both topological structure and nodal attributes. The encoder of this unsupervised approach encodes the attribute information using the output GCN embedding, while the decoder reconstructs both the structure and attribute information using non-linearly transformed embeddings of the output GCN layer. The basic idea behind anomaly detection methods based on reconstruction errors is that the normal instances can be reconstructed with small errors, while anomalous instances are often reconstructed with large errors. More recently, Kumagai *et al.* [15] have proposed two GCN-based models for semi-supervised anomaly detection. The first model uses only labeled normal instances, whereas the second one employs labeled normal and anomalous instances. Both models are trained to minimize the volume of a hypersphere that encloses the GCN-learned node embeddings of normal instances, while embedding the anomalous ones outside the hypersphere.

Inspired by the implicit fairing concept in geometry processing for triangular mesh smoothing [16], we introduce a graph fairing convolutional network architecture, which we call GFCN, for deep semi-supervised anomaly detection. In addition to performing graph convolution, GFCN uses a skip connection to combine both the initial node representation and the aggregated node neighborhood representation, enabling it to memorize information across distant nodes. While most graph convolutions with skip connections are based on heuristics, GFCN is theoretically motivated by implicit fairing and derived from the Jacobi iterative method. In contrast to GCN-based methods that use a first-order approximation of spectral graph convolutions and a renormalization trick in their layer-wise propagation rules to avoid numerical instability, our GFCN model does not require any renormalization, while still maintaining the key property of convolution as a neighborhood aggregation operator. Hence, repeated application of the GFCN's layer-wise propagation rule provides a computationally efficient convolutional process, leading to numerical stability while avoiding the issue of exploding/vanishing gradients. The proposed framework achieves better anomaly detection performance, as GFCN uses a multi-layer architecture, together with skip connections, and non-linear activation functions to extract high-order information of graphs as discriminative features. Multi-layer architectures enable the model to learn hierarchical representations of the graph, where lower layers capture lower-level features and higher layers capture higher-level abstractions. Skip connections allow information to bypass intermediate layers and preserve low-level details, improving the flow of information and preventing vanishing gradients, and more importantly leading to more accurate representations, thereby yielding a more effective detection of anomalies. Moreover, GFCN inherits all benefits of GCNs, including accuracy, efficiency and ease of training.

In addition to capturing information from distant graph nodes through skip connections between layers, the proposed GFCN model is flexible and exploits both the graph structure and node features for learning discriminative node representations in an effort to detect anomalies in a semi-supervised setting. Not only does GFCN outperforms strong anomaly detection baselines, but it is also surprisingly simple, yet very effective at identifying anomalies. The main contributions of this work can be summarized as follows:

- We propose a novel multi-layer graph convolutional network with a skip connection for semi-supervised anomaly detection by effectively exploiting both the graph structure and attribute information.

- We introduce a learnable skip-connection module, which helps nodes propagate through the network's layers and hence substantially improves the quality of the learned node representations.

- We analyze the complexity of the proposed model and train it on a regularized, weighted cross-entropy loss function by leveraging unlabeled instances to improve performance.

- We demonstrate through extensive experiments that our model can capture the anomalous behavior of graph nodes, leading to state-of-the-art performance across several benchmark datasets.

The rest of this paper is organized as follows. In Section 2, we review important relevant work. In Section 3, we outline the background for spectral graph theory and present the problem formulation. In Section 4, we introduce a graph convolutional network architecture with skip connection for deep semi-supervised anomaly detection. In Section 5, we present experimental results to demonstrate the competitive performance of

our approach on five standard benchmarks. Finally, we conclude in Section 6 and point out future work directions.

2 Related Work

The basic goal of anomaly detection is to identify abnormal instances, which do not conform to the expected pattern of other instances in a dataset. To achieve this goal, various anomaly detection techniques have been proposed, which can distinguish between normal and anomalous instances [1]. Most mainstream approaches are one-class classification models [4, 5] or graph-based anomaly methods [13].

Deep Learning for Anomaly Detection. While shallow methods such as one-class classification models require explicit hand-crafted features, much of the recent work in anomaly detection leverages deep learning [2], which has shown remarkable capabilities in learning discriminative feature representations by extracting high-level features from data using multilayered neural networks. Ruff *et al.* [9] develop a deep SVDD anomaly detection framework, which is basically an extension of the shallow one-class classification SVDD approach. The basic idea behind SVDD is to find the smallest hypersphere that contains all instances, except for some anomalies. Deep SVDD is an unsupervised learning model that learns to extract the common factors of variation of the data distribution by training a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data. The centroid of the hypersphere is usually set to the mean of the feature representations learned by performing a single initial forward pass. In order to improve model performance, Ruff *et al.* [10] propose Deep SAD, a generalization of the unsupervised Deep SVDD to the semi-supervised setting. The key difference between these two deep anomaly detection models is that the objective function of Deep SAD also includes a loss term for labeled normal and anomalous instances. The idea behind this loss term is to minimize (resp. maximize) the squared Euclidean distance between the labeled normal (resp. anomalous) instances and the hypersphere centroid. However, both Deep SVDD and Deep SAD suffer from the hypersphere collapse problem due to the learning of a trivial solution. In other words, the network's learned features tend to converge to the centroid of the hypersphere if no constraints are imposed on the architectures of the models. Cevikalp *et al.* [17] considered the hypersphere centers as parameters that can be learned and updated according to the evolving deep feature representations. Another line of work uses deep generative models to address the anomaly detection problem [18, 19]. These generative networks are able to localize anomalies, particularly in images, by simultaneously training a generator and a discriminator, enabling the detection of anomalies on unseen data based on unsupervised training of the model on anomaly-free data [20]. However, the use of deep generative models in anomaly detection has been shown to be quite problematic and unintuitive, particularly on image data [21].

Graph Convolutional Networks for Anomaly Detection. GCNs have recently become the de facto model for learning representations on graphs, achieving state-of-the-art perfor-

mance in various application domains, including anomaly detection [14, 15]. Ding *et al.* [14] present an unsupervised graph anomaly detection framework using a GCN-based autoencoder. This approach leverages both the topological structure and nodal attributes, with an encoder that maps the attribute information into a low-dimensional feature space and a decoder that reconstructs the structure as well as the attribute information using the learned latent representations. The basic idea behind this GCN-based autoencoder is that the normal instances can be reconstructed with small errors, while anomalous instances are often reconstructed with large errors. However, methods based on reconstruction errors are prone to outliers and often require noise-free data for training. On the other hand, some of the main challenges associated with graph anomaly detection is the lack of labeled graph nodes (i.e. no information is available about which instances are actually anomalous and which ones are normal) and data imbalance, as abnormalities occur rarely and hence a tiny fraction of instances is expected to be anomalous. To circumvent these issues, Kumagai *et al.* [15] propose two semi-supervised anomaly detection models using GCNs for learning latent representations. The first model uses only labeled normal instances, while the second one employs both labeled normal and anomalous instances. However, both models are trained to minimize the volume of a hypersphere that encloses the GCN-learned node embeddings of normal instances, and hence they also suffer from the hypersphere collapse problem. By contrast, our semi-supervised GFCN model does not suffer from the above mentioned issues. In addition to leveraging the graph structure and node attributes, GFCN learns from both labeled and unlabeled data in order to improve model performance.

Graph Neural Networks with Skip Connections. Despite the success of GNN-based models in learning node representations, they are prone to over-smoothing, which can negatively impact their performance. Over-smoothing occurs when stacking multiple graph convolution layers causes node representations to become indistinguishable, leading to a loss of valuable information. To tackle the over-smoothing problem, several approaches that leverage skip connections have been proposed. Skip connections can be categorized into four main types: residual connections, initial connections, jumping connections, and dense connections [22]. JK-Net [23] uses jumping knowledge network connections to connect each layer to the last one, maintaining the feature mappings in lower layers. APPNP [24], which approximate PageRank with power iteration, uses initial connection by connecting each layer to the original feature matrix. By decoupling feature transformation and propagation, APPNP can aggregate information from multi-hop neighbors without increasing the number of layers in the network. GCNII [25] employs initial residual and identity mapping to mitigate the over-smoothing problem. At each layer, the initial residual constructs a skip connection from the input layer, while the identity mapping adds an identity matrix to the weight matrix. ResGCN [26] is a residual graph convolutional network that extends the depth of GCNs by using residual/dense connections and dilated convolutions. In our proposed GFCN model, we apply a skip connection that reuses the initial node features at each layer with the goal of combining both the aggregated

node neighborhood representation and the initial node representation. While most graph convolutions with skip connections are based on heuristics, our GFCN model is theoretically motivated by implicit fairing and its layerwise propagation rule is derived from the iterative solution of the implicit fairing equation via the Jacobi method.

3 Preliminaries and Problem Statement

We introduce our notation and present a brief background on spectral graph theory [27–29], followed by our problem formulation of semi-supervised anomaly detection on graphs.

Basic Notions. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The graph structure is encoded by an $N \times N$ adjacency matrix $\mathbf{A} = (\mathbf{A}_{ij})$ whose (i, j)-th entry is equal to the weight of the edge between neighboring nodes i and j, and 0 otherwise. We also denote by $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^\intercal$ an $N \times F$ feature matrix of node attributes, where $\mathbf{x}_i$ is an F-dimensional row vector for node i. This real-valued feature vector is often referred to as a graph signal, which assigns a value to each node in the graph.

Spectral Graph Theory. The normalized Laplacian matrix is defined as

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}, \tag{1}$$

where $\mathbf{D} = \mathrm{diag}(\mathbf{A1})$ is the diagonal degree matrix, and $\mathbf{1}$ is an N-dimensional vector of all ones. Since the normalized Laplacian matrix is symmetric positive semi-definite, it admits an eigendecomposition given by $\mathbf{L} = \mathbf{U\Lambda U}^\intercal$, where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_N)$ is an orthonormal matrix whose columns constitute an orthonormal basis of eigenvectors and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix comprised of the corresponding eigenvalues such that $0 = \lambda_1 \leq \cdots \leq \lambda_N \leq 2$. If $\mathcal{G}$ is a bipartite graph, then the spectral radius (i.e. largest absolute value of all eigenvalues) of the normalized Laplacian matrix is equal to 2. The normalized Laplacian matrix has eigenvalues in the range [0,2], which makes spectral graph analysis algorithms more stable and reliable compared to algorithms that use the unnormalized Laplacian matrix with eigenvalues that can be much larger. Moreover, scaling by the inverse square root of the degree matrix helps reduce the influence of highly connected nodes.

Problem Statement. Anomaly detection aims at identifying anomalous instances, which do not conform to the expected pattern of other instances in a dataset. It differs from binary classification in that it distinguishes between normal and anomalous observations. Also, the distribution of anomalies is not usually known a priori.

Let $\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_l}$ be a set of labeled data points $\mathbf{x}_i \in \mathbb{R}^F$ and their associated known labels $y_i \in \{0, 1\}$ with 0 and 1 representing "normal" and "anomalous" observations, respectively, and $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=N_l+1}^{N_l+N_u}$ be a set of unlabeled data points, where $N_l + N_u = N$. Hence, each node i can be labeled with a 2-dimensional one-hot encoding vector $\mathbf{y}_i = (y_i, 1 - y_i)$.

The goal of semi-supervised anomaly detection on graphs is to estimate the anomaly scores of the unlabeled graph nodes.

Nodes with high anomaly scores are considered anomalous, while nodes with lower scores are deemed normal.

4 Proposed Method

In this section, we begin by succinctly summarizing some of the most common spectral filters on graphs. Then, we propose a graph convolutional network with skip connection using the concept of implicit fairing on graphs. In particular, we examine the main components of the proposed architecture and analyze the complexity of the model. In addition, we introduce an anomaly scoring function defined in terms of the weighted cross-entropy between the ground-truth labels of the graph test nodes and the model's predicted probabilities.

4.1 Spectral Graph Filtering

The idea of spectral filtering on graphs was first introduced in [30] in the context of triangular mesh smoothing. The goal of spectral graph filtering is to use polynomial or rational polynomial filters defined as functions of the graph Laplacian (or equivalently its eigenvalues) in an effort to attenuate high-frequency noise corrupting the graph signal. These functions are usually referred to as frequency responses or transfer functions. While polynomial filters have finite impulse responses, their rational counterparts have infinite impulse responses. Despite the fact that the Laplacian matrix is commonly used in spectral graph theory, it does not, however, provide a natural way to normalize the frequency domain representation of a graph signal, which can lead to scaling and convergence issues in spectral graph filtering. In contrast, the normalized Laplacian matrix provides a way to normalize the frequency domain representation of a graph signal, which can improve the stability and convergence properties of spectral graph filtering. Specifically, the normalized Laplacian matrix is scaled by the inverse square root of the degree matrix, which helps normalize the contributions of each node's neighbors to the overall graph signal. Applying a spectral graph filter with transfer function h on the graph signal $\mathbf{X} \in \mathbb{R}^{N \times F}$ yields

$$\mathbf{H} = h(\mathbf{L})\mathbf{X} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\intercal \mathbf{X} = \mathbf{U}\,\mathrm{diag}(h(\lambda_i))\mathbf{U}^\intercal \mathbf{X}, \tag{2}$$

where $\mathbf{H}$ is the filtered graph signal. However, this filtering process necessitates the computation of the eigenvalues and eigenvectors of the Laplacian matrix, which is prohibitively expensive for large graphs. To circumvent this issue, spectral graph filters are usually approximated using Chebyshev polynomials [11, 31, 32] or rational polynomials [33–35].

4.2 Implicit Fairing

Graph fairing refers to the process of designing and computing smooth graph signals on a graph in order to filter out undesirable high-frequency noise while retaining the graph geometric features as much as possible. The implicit fairing method, which uses implicit integration of a diffusion process for graph filtering, has shown to allow for both efficiency and stability [16].

The implicit fairing filter is an infinite impulse response filter whose transfer function is given by $h_s(\lambda) = 1/(1 + s\lambda)$, where s is a positive parameter. Hence, performing graph filtering with implicit fairing is equivalent to solving the following sparse linear system:

$$(\mathbf{I} + s\mathbf{L})\mathbf{H} = \mathbf{X}, \tag{3}$$

which we refer to as implicit fairing equation. It is worth pointing out that this equation can also be obtained by minimizing the following objective function

$$\mathcal{J}(\mathbf{H}) = \frac{1}{2}\|\mathbf{H} - \mathbf{X}\|_F^2 + \frac{s}{2}\operatorname{tr}(\mathbf{H}^\mathsf{T}\mathbf{L}\mathbf{H}), \tag{4}$$

where $\|\cdot\|_F$ and $\operatorname{tr}(\cdot)$ denote the Frobenius norm and trace operator, respectively.

The implicit fairing filter enjoys several nice properties, including unconditional stability as $h_s(\lambda)$ is always in $[0, 1]$, and also preservation of the average value (i.e. DC value or centroid) of the graph signal as $h_s(0) = 1$ for all s. As shown in Figure 1, the higher the value of the scaling parameter, the closer the implicit fairing filter becomes to the ideal low-pass filter.
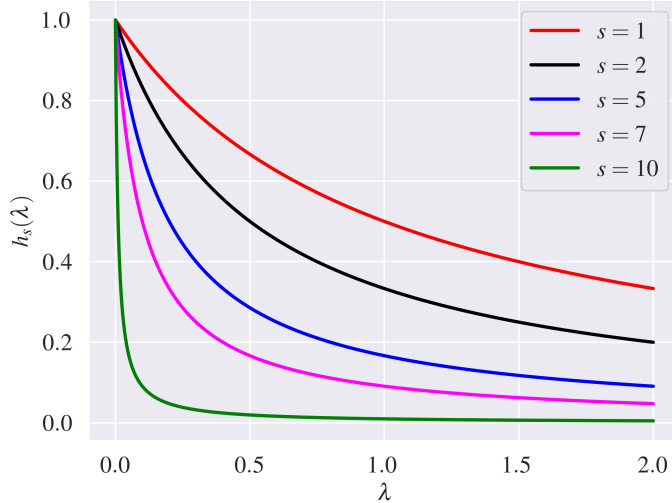

Figure 1: Transfer function of the implicit fairing filter for various values of the scaling parameter.

4.3   Spectral Analysis

The matrix $\mathbf{I} + s\mathbf{L}$ is symmetric positive definite with minimal eigenvalue equal to 1 and maximal eigenvalue bounded from above by $1 + 2s$. Hence, the condition number κ of $\mathbf{I} + s\mathbf{L}$ satisfies

$$\kappa \le 1 + 2s, \tag{5}$$

where κ, which is defined as the ratio of the maximum to minimum stretching, is also equal to the maximal eigenvalue of $\mathbf{I} + s\mathbf{L}$. Intuitively, the condition number measures how much can a change (i.e. small perturbation) in the right-hand side of the implicit fairing equation affects the solution. In fact, it can be readily shown that the resulting relative change in the solution of the implicit fairing equation is bounded from above by the condition number multiplied by the relative change in the right-hand side.

4.4   Iterative Solution

One of the simplest iterative techniques for solving a matrix equation is Jacobi's method, which uses matrix splitting. Since the matrix $\mathbf{I} + s\mathbf{L}$ can be split into the sum of a diagonal matrix and an off-diagonal matrix

$$\mathbf{I} + s\mathbf{L} = (1 + s)\mathbf{I} - s\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{6}$$

the implicit fairing equation can then be solved iteratively using the Jacobi method as follows:

$$\mathbf{H}^{(t+1)} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^{(t)}\mathbf{\Theta}_s + \mathbf{X}\widetilde{\mathbf{\Theta}}_s, \tag{7}$$

where $\mathbf{\Theta}_s = \operatorname{diag}(s/(1+s))$ and $\widetilde{\mathbf{\Theta}}_s = \operatorname{diag}(1/(1+s))$ are $F \times F$ diagonal matrices, each of which has equal diagonal entries, and $\mathbf{H}^{(t)}$ is the t-th iteration of $\mathbf{H}$. Since the spectral radius of the normalized adjacency matrix is equal to 1, it follows that the spectral radius of the Jacobi's iteration matrix

$$\mathbf{C} = \frac{s}{1 + s}\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{8}$$

is equal to $s/(1+s)$, which is always smaller than 1. Hence, the convergence of the iterative method given by Eq. (7) holds.

4.5   Graph Fairing Convolutional Network

At the core of graph representation learning is the concept of propagation rule, which determines how information is passed between nodes in a graph. It involves updating the current node features by aggregating information from their neighboring nodes, followed by a non-linear activation function to produce an updated representation for the node. Inspired by the Jacobi iterative solution of the implicit fairing equation, we propose a multi-layer graph fairing convolutional network (GFCN) with the following layer-wise propagation rule:

$$\mathbf{H}^{(\ell+1)} = \sigma(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^{(\ell)}\mathbf{\Theta}^{(\ell)} + \mathbf{X}\widetilde{\mathbf{\Theta}}^{(\ell)}), \tag{9}$$

where $\mathbf{\Theta}^{(\ell)}$ and $\widetilde{\mathbf{\Theta}}^{(\ell)}$ are learnable weight matrices, $\sigma(\cdot)$ is an element-wise activation function, $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times F_\ell}$ is the input feature matrix of the ℓ-th layer with F_ℓ feature maps for $\ell = 0, \ldots, L - 1$. The input of the first layer is the initial feature matrix $\mathbf{H}^{(0)} = \mathbf{X}$.

Note that in addition to performing graph convolution, which essentially averages the features of the immediate (i.e. first-order or 1-hop) neighbors of nodes, the layer-wise propagation rule of GFCN also applies a skip connection that reuses the initial node features, as illustrated in Figure 2. In other words, GFCN combines both the aggregated node neighborhood representation and the initial node representation, hence memorizing information across distant nodes. While most of the existing graph convolutions with skip connections [23, 34] are based on heuristics, our graph fair convolution is theoretically motivated by implicit fairing and derived directly from the Jacobi iterative method. It is important to mention that the convolution operation in the proposed propagation rule involves aggregating information from a node's neighbors and combining it with the
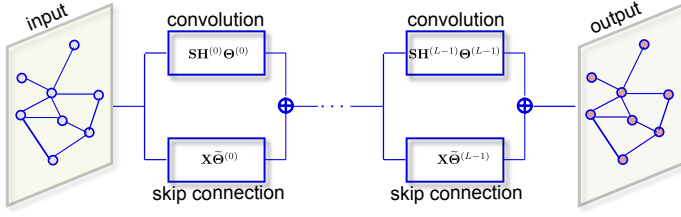
Figure 2: Schematic layout of the proposed GFCN architecture. Each block comprises a graph convolution and a skip connection, followed by an activation function, where $\mathbf{S}$ denotes the normalized adjacency matrix. The GFCN model takes as input the adjacency matrix $\mathbf{A}$ and initial feature matrix $\mathbf{X} = \mathbf{H}^{(0)}$. At each layer, a node aggregates information from its neighboring nodes and the initial feature matrix through skip connection. The aggregated information is then transformed using learnable weight matrices. The resulting representation is then passed to the next layer for further propagation. Finally, the output is the latent graph representation $\mathbf{H}^{(L)}$ from the last network layer.

node's own features to produce a new set of features. As a result, the normalized adjacency matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is particularly useful in this context because it provides a way to normalize the aggregation of neighbor features, which helps avoid the problem of over-reliance on highly connected nodes. This normalization also helps make the convolution operation more stable and better conditioned, which can improve the convergence and generalization performance of our mode. The normalized adjacency matrix considers both the number of neighbors connected to a node and the number of neighbors connected to each of those neighboring nodes.

In contrast to GCN-based methods which use a first-order approximation of spectral graph convolutions and a renormalization trick in their layer-wise propagation rules to avoid numerical instability, our GFCN model does not require any renormalization as the spectral radius of the normalized adjacency matrix is equal to 1, while still maintaining the key property of convolution as a neighborhood aggregation operator. Hence, repeated application of the GFCN's layer-wise propagation rule provides a computationally efficient convolutional process, leading to numerical stability and avoidance of exploding/vanishing gradients. It should also be pointed out that similar to GCN, the proposed GFCN model also does not require explicit computation of the Laplacian eigenbasis. In addition, the aggregation of GFCN with skip connections between the initial feature matrix and each hidden layer does not require filtered learning of the hidden layers. Skip connections are a mechanism that allows our model to directly pass on the information from the initial feature matrix (input) to the deeper hidden layers, thereby helping to retain the original node features to prevent loss of important information during the aggregation process.

4.6 Model Prediction

The layer-wise propagation rule of GFCN is basically a node embedding transformation that projects both the input $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times F_\ell}$ into a trainable weight matrix $\mathbf{\Theta}^{(\ell)} \in \mathbb{R}^{F_\ell \times F_{\ell+1}}$ and the initial feature matrix $\mathbf{X}$ into the skip-connection weight matrix $\widetilde{\mathbf{\Theta}}^{(\ell)} \in \mathbb{R}^{F \times F_{\ell+1}}$, with $F_{\ell+1}$ feature maps. Then, a point-wise activation function $\sigma(\cdot)$ such as $\text{ReLU}(\cdot) = \max(0, \cdot)$ is applied to obtain the output node embedding matrix. The aggregation scheme of GFCN is depicted in Figure 3. Note that GFCN uses skip connections between the initial feature matrix and each hidden layer. Skip connections not only allow the model to carry over information from the initial node attributes, but also help facilitate training of multi-layer networks.
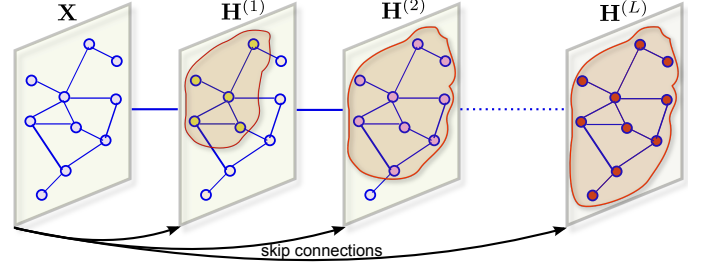

Figure 3: Illustration of the GFCN aggregation scheme with skip connections.

The embedding $\mathbf{H}^{(L)}$ of the last layer of GFCN contains the final output node embeddings, which can be used as input for downstream tasks such as graph classification, clustering, visualization, link prediction, recommendation, node classification, and anomaly detection. Since anomaly detection can be posed as a binary classification problem, we apply the point-wise softmax function (i.e. sigmoid in the binary case) to obtain an $N \times 2$ matrix of predicted labels for graph nodes

$$\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_N)^\mathsf{T} = \text{softmax}(\mathbf{H}^{(L)}), \qquad (10)$$

where $\hat{\mathbf{y}}_i = (\hat{y}_i, 1 - \hat{y}_i)$ is a two-dimensional raw vector of predicted probabilities, with $\hat{y}_i$ the probability that the network associates the i-th node with one of the two classes (i.e. 1 for anomalous and 0 for normal).

4.7 Model Complexity

For simplicity, we assume the feature dimensions are the same for all layers, i.e. $F_\ell = F$ for all ℓ, with $F \ll N$. The time complexity of an L-layer GFCN is $\mathcal{O}(L\|\mathbf{A}\|_0 F + LNF^2)$, where $\|\mathbf{A}\|_0$ denotes the number of non-zero entries of the sparse adjacency matrix. Note that multiplying the normalized adjacency matrix with an embedding costs $\mathcal{O}(\|\mathbf{A}\|_0 F)$ in time, while multiplying an embedding with a weight matrix costs $\mathcal{O}(NF^2)$. Also, multiplying the initial feature matrix by the skip-connection weight matrix costs $\mathcal{O}(NF^2)$.

For memory complexity, an L-layer GFCN requires $\mathcal{O}(LNF + LF^2)$ in memory, where $\mathcal{O}(LNF)$ is for storing all embeddings and $\mathcal{O}(LF^2)$ is for storing all layer-wise weight matrices.

Therefore, our proposed GFCN model has the same time and memory complexity as GCN, albeit GFCN takes into account distant graph nodes for improved learned node representations.

4.8 Model Training

The parameters (i.e. weight matrices for different layers) of the proposed GFCN model for semi-supervised anomaly detection are learned by minimizing the following regularized loss function

$$\mathcal{L} = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathcal{C}_\alpha(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \frac{\beta}{2} \sum_{\ell=0}^{L-1} \left(\|\mathbf{\Theta}^{(\ell)}\|_F^2 + \|\widetilde{\mathbf{\Theta}}^{(\ell)}\|_F^2 \right),$$
(11)

where $\mathcal{C}_\alpha(\mathbf{y}_i, \hat{\mathbf{y}}_i)$ is the weighted cross-entropy given by

$$\mathcal{C}_\alpha(\mathbf{y}_i, \hat{\mathbf{y}}_i) = -\alpha \, y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i),$$
(12)

which measures the dissimilarity between the one-hot encoding vector $\mathbf{y}_i$ of the ith node and the corresponding vector $\hat{\mathbf{y}}_i$ of predicted probabilities. This dissimilarity decreases as the value of the predicted probability approaches the ground-truth label. The weight parameter α adjusts the importance of the positive labels by assigning more weight to the anomalous class, while the parameter β controls the importance of the regularization term, which is added to prevent overfitting. The regularization term is the sum of the squared elements of the learnable weight matrices for each layer. It is important to mention that we only use the normal class labeled instances to train our model.

We optimize our model using the Adam optimizer [36], which is a modified version of Stochastic Gradient Descent (SGD) that uses adaptive moment estimation. The intuition behind the use of the weighted cross-entropy loss function is to assign a higher weight to the anomalous nodes than to the normal nodes, so that the model is encouraged to correctly identify the anomalous nodes even if they are rare and overshadowed by the large number of normal nodes. The regularization term, on the other hand, penalizes large weight values in the learnable weight matrices, which helps to reduce the complexity of the model and improve its generalization performance. The strength of the regularization is controlled by the value of the hyperparameter β, which is tuned using grid search.

The weight matrices of our GFCN model are initialized randomly with small values using a normal distribution to ensure that the variance of the activations and gradients is roughly the same across all layers of the network. During training, the optimizer adjusts the weight matrices to minimize the regularized loss function. The training process involves choosing the hyperparameters, computing the regularized weighted cross-entropy loss, feeding forward and backpropagating the inputs, and updating the weight matrices using the Adam optimizer. This process is repeated for multiple epochs until the model converges or the validation loss does not decrease after a specified number of consecutive epochs.

4.9 Model Inference

Once the model is trained, we can use the weighted cross-entropy errors to assess the abnormality of nodes. To this end, we define the anomaly score of the ith test node as

$$s_i = \mathcal{C}_\alpha(\mathbf{y}_i, \hat{\mathbf{y}}_i).$$
(13)

Since the range of the weighted cross-entropy is $[0, \infty)$ (e.g. infinite value when $y_i = 1$ and $\hat{y}_i = 0$), we apply min-max normalization to bring all anomaly scores into the range [0,1] as follows:

$$\tilde{s}_i = \frac{s_i - s_{\min}}{s_{\max} - s_{\min}},$$
(14)

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum, respectively, of the anomaly scores in the test set. Nodes with scores larger than a certain threshold are considered anomalies. Hence, we can compute a ranked list of anomalies according to their normalized anomaly scores. In other words, we compute the anomaly scores of each node in the test set, and then the top-r nodes with higher scores are identified as anomalies for a user-specified value of r.

5 Experiments

In this section, we conduct extensive experiments to assess the performance of the proposed anomaly detection framework in comparison with state-of-the-art methods on several benchmark datasets. The source code to reproduce the experimental results is made publicly available on GitHub[1].

5.1 Datasets

We demonstrate and analyze the performance of the proposed model on three citation networks: Cora, Citeseer, and Pubmed [37], and two co-purchase graphs: Amazon Photo and Amazon Computers [38]. The summary descriptions of these benchmark datasets are as follows:

- Cora is a citation network dataset consisting of 2708 nodes representing scientific publications and 5429 edges representing citation links between publications. All publications are classified into 7 classes (research topics). Each node is described by a binary feature vector indicating the absence/presence of the corresponding word from the dictionary, which consists of 1433 unique words.

- Citeseer is a citation network dataset composed of 3312 nodes representing scientific publications and 4723 edges representing citation links between publications. All publications are classified into 6 classes (research topics). Each node is described by a binary feature vector indicating the absence/presence of the corresponding word from the dictionary, which consists of 3703 unique words.

- Pubmed is a citation network dataset containing 19717 scientific publications pertaining to diabetes and 44338 edges representing citation links between publications. All publications are classified into 3 classes. Each node is described by a TF/IDF weighted word vector from the dictionary, which consists of 500 unique words.

- Amazon Computers and Amazon Photo datasets are co-purchase graphs [38], where nodes represent goods and

---

[1] https://github.com/MahsaMesgaran/GFCN

7

edges indicate that two goods are frequently bought together. The node features are bag-of-words encoded product reviews, while the class labels are given by the product category.

- ogbn-arxiv dataset is a large-scale graph dataset from open graph benchmark (OGB) representing the citation network between all computer science (CS) arXiv papers. In this dataset over 169k nodes and 1.1m edges are contained. Each node has a 128-dimensional feature vector obtained by averaging the embeddings of words in the article's title and abstract.

Since there is no ground truth of anomalies in these datasets, we employ the commonly-used protocol [15] in anomaly detection by treating the smallest class for each dataset as the anomaly class and the remaining classes as the normal class. Dataset statistics are summarized in Table 1, where anomaly rate refers to the percentage of abnormalities in each dataset. For all datasets, we only use the normal class labeled instances to train the model.

Table 1: Summary statistics of datasets.

Dataset	Nodes	Edges	Features	Classes	Anomaly Rate (%)
Cora	2708	5278	1433	7	0.06
Citeseer	3327	4732	3703	6	0.07
Pubmed	19717	44338	500	3	0.21
Photo	7487	119043	745	8	0.04
Computers	13381	245778	767	10	0.02
ogbn-arxiv	169343	1166243	128	40	0.02

5.2 Baseline Methods

We evaluate the performance of the proposed method against various baselines, including one-class support vector machines (OC-SVMs) [4], imbalanced vertex diminished (ImVerde) [39], one-class deep support vector data description (Deep SVDD) [9], deep anomaly detection on attributed networks (Dominant) [14], one-class deep semi-supervised anomaly detection (Deep SAD) [10], graph convolutional networks (GCNs) [12], GCN-based anomaly detection (GCN-N and GCN-AN) [15], graph random neural networks (GRAND) [40], semi-Supervised node classification on graph with few labels via non-parametric distribution assignment (GraFN) [41], and re-weighting the influence of labeled nodes (ReNode) [42]. For baselines, we mainly consider methods that are closely related to GFCN and/or the ones that are state-of-the-art anomaly detection frameworks. A brief description of these strong baselines can be summarized as follows:

- **OC-SVM** [4] is an unsupervised one-class anomaly detection technique, which learns a discriminative hyperplane boundary around the normal instances using support vector machines by maximizing the distance from this hyperplane to the origin of the high-dimensional feature space.

- **ImVerde** [39] is a semi-supervised graph representation learning technique for imbalanced graph data based on a variant of random walks by adjusting the transition probability each time a graph node is visited by the random particle.

- **Deep SVDD** [9] is an unsupervised anomaly detection method, inspired by kernel-based one-class classification and minimum volume estimation, which learns a spherical, instead of a hyperplane, boundary in the feature space around the data using support vector data description. It trains a deep neural network while minimizing the volume of a hypersphere that encloses the network embeddings of the data. Normal instances fall inside the hypersphere, while anomalies fall outside.

- **Dominant** [14] is a deep autoencoder based on GCNs for unsupervised anomaly detection on attributed graphs. It employs an objective function defined as a convex combination of the reconstruction errors of both graph structure and node attributes. These learned reconstruction errors are then used to assess the abnormality of graph nodes.

- **Deep SAD** [10] is a semi-supervised anomaly detection technique, which generalizes the unsupervised Deep SVDD approach to the semi-supervised setting by incorporating a new term for labeled training data into the objective function. The weights of the Deep SAD network are initialized using an autoencoder pre-training mechanism.

- **GCN** [12] is a deep graph neural network for semi-supervised learning of graph representations, encoding both local graph structure and attributes of nodes. It is an efficient extension of convolutional neural networks to graph-structured data, and uses a graph convolution that aggregates and transforms the feature vectors from the local neighborhood of a graph node.

- **GCN-N** and **GCN-AN** [15] are GCN-based, semi-supervised anomaly detection frameworks, which rely on minimizing the volume of a hypersphere that encloses the node embeddings to detect anomalies. Node embeddings placed inside and outside this hypersphere are deemed normal and anomalous, respectively. GCN-N uses only normal label information, while GCN-AN uses both anomalous and normal label information.

- **GRAND** [40] is a semi-supervised learning on graphs when labeled nodes are scarce. This technique relies on a random propagation strategy to perform graph data augmentation and employs consistency regularization to optimize prediction consistency of unlabeled nodes across different data augmentations.

- **GraFN** [41] is a semi-supervised node representation learning for graphs with few labeled nodes. This technique exploits the self-supervised loss to ensure nodes that belong to the same class to be grouped together on differently augmented graphs. GraFN randomly samples support nodes from the labeled nodes and anchor nodes from the entire

graph, and non-parametrically compute two predicted class distributions from two augmented graphs based on the anchor supports similarity.

- **ReNode** [42] is a semi-supervised node classification technique addressing the the topology-imbalance node representation learning as a graph specific imbalance learning problem. To measure the degree of topology imbalance, a conflict detection-based metric, Totoro, is used to locate node position. ReNode adjusts the training weights of labeled nodes based on their topological positions.

5.3 Evaluation Metric

In order to evaluate the performance of our proposed framework against the baseline methods, we use AUC, the area under the receiving operating characteristic (ROC) curve, as a metric. AUC summarizes the information contained in the ROC curve, which plots the true positive rate versus the false positive rate, at various thresholds [43, 44]. Larger AUC values indicate better performance at distinguishing between anomalous and normal instances. An uninformative anomaly detector has an AUC equal to 50% or less. An AUC of 50% corresponds to a random detector (i.e. for every correct prediction, the next prediction will be incorrect), whereas an AUC score smaller than 50% indicates that a detector performs worse than a random detector.

5.4 Implementation Details

For fair comparison, we implement the proposed method and baselines in PyTorch using the PyTorch Geometric library. Following common practices for evaluating performance of GCN-based models [12, 15], we train our 2-layer GFCN model for 100 epochs using the Adam optimizer [36] with a learning rate of 0.1. We tune the latent representation dimension by hand, and set it to 128. The hyperparameters α and β are chosen via grid search with cross-validation over the sets $\{10^{-4}, 10^{-3}, \ldots, 1\}$ and $\{2, 3, \ldots, 10\}$, respectively. We tune hyperparameters using the validation set, and terminate training if validation loss does not decrease after 10 consecutive epochs. For each dataset, we consider the settings where 2.5%, 5% and 10% of instances are labeled, and we compute the average and standard deviation of test AUCs over ten runs.

5.5 Anomaly Detection Performance

Tables 2-4 present the anomaly detection results on the five datasets. The best results are highlighted as bold. For each dataset, we report the AUC averaged over 10 runs as well as the standard deviation, at various ratios of labeled instances. As can be seen, our GFCN model consistently achieves the best performance on all datasets, except in the case of the Cora dataset when 10% of instances are labeled. In that case, GCN-AN yields a marginal improvement of 0.7% over GFCN, despite the fact that GCN-AN is trained on both normal and anomalous instances, whereas our model is trained only on normal instances. In addition, ImVerde, Deep SAD, GCN and GCN-AN all perform reasonably well on all datasets at various levels of label

rates, but we find that GFCN outperforms these baselines on almost all datasets, while being considerably simpler. An AUC score of 50% or less indicates that the baseline is an uninformative anomaly detector.

On the Amazon Computers dataset, Table 2 shows that the proposed GFCN approach performs on par with GCN-AN, but outperforms all baselines on the other four datasets. In particular, GFCN yields 16.2% and 15.7% performance gains over Deep SAD on the Cora and Photo datasets, respectively. These gains are consistent with the results shown in Tables 3-4. We argue that the better performance of GFCN over GCN-N and Deep SAD is largely attributed to the fact that our model does not suffer from the hypersphere collapse problem. Interestingly, the performance gains are particularly higher at the lower label rate 2.5%, confirming the usefulness of semi-supervised learning in that it improves model performance by leveraging unlabeled data. Another interesting observation is that in general both GCN-AN and GCN-N yield relatively high AUC standard deviations compared to our GFCN model, indicating that our model has less variability than these two strong baselines.

Lastly, we examined the training times (in seconds) for GFCN on Cora when 10% of all instances were labeled. We also recorded the training times of GFCN, GCN-AN, GCN-N, and GCN on Cora for which we obtained 3.19, 4.12, 2.31, and 2.06 seconds, respectively. Since GFCN uses the skip connection, it took more training time than GCNs. However, the experiment shows the proposed method could learn the abnormalities fast enough.

5.6 Parameter Sensitivity Analysis

The weight hyperparameter α of the weighted cross-entropy and the regularization hyperparameter β play an important role in the anomaly detection performance of the proposed GFCN framework. We conduct a sensitivity analysis to investigate how the performance of GFCN changes as we vary these two hyperparameters. In Figure 4, we analyze the effect of the hyperparameter α by plotting the AUC results of GFCN vs. α using various label rates for all datasets, where α varies from 2 to 10. We can see that with a few exceptions, our model generally benefits from relatively larger values of the weight hyperparameter. For almost all datasets, our model achieves satisfactory performance with $\alpha = 4$.

In Figure 5, we plot the average AUCs, along with the standard error bars, of our GFCN model vs. β using various label rates for all datasets, and by varying the value of β from 10^{-4} to 1. Notice that the best performance is generally achieved when $\beta = 0.01$, except in the cases of the Citeseer and Pubmed datasets, on which the best performance is obtained when $\beta = 0.1$. In general, when the regularization hyperparameter increases, the performance improves rapidly at the very beginning, but then deteriorates after reaching the best setting due to overfitting. An interesting observation is that GFCN generally shows steady increase in performance with the regularization parameter, except on the Citeseer dataset when the label rate is 5%, whereas the performance on the other datasets degrades after reaching a certain threshold.

Table 2: Test AUC (%) averaged over 10 runs when 2.5% of instances are labeled. We also report the standard deviation. Boldface numbers indicate the best anomaly detection performance.

	Dataset					
Method	Cora	Citeseer	Pubmed	Photo	Computers	ogbn-arxiv
OC-SVM [4]	50.0±0.1	50.6±0.4	68.9±0.9	51.9±0.6	47.3±0.7	-
ImVerde [39]	85.9±6.1	60.3±6.5	94.3±0.5	89.1±1.4	98.5±0.7	-
Deep SVDD [9]	69.6±6.5	55.3±1.6	73.7±6.3	52.3±1.4	46.6±1.5	-
Dominant [14]	52.3±0.9	53.9±0.6	50.8±0.4	38.1±0.4	46.8±1.2	-
Deep SAD [10]	72.7±6.0	53.8±2.9	91.3±2.4	81.9±5.7	92.2±2.5	-
GCN [12]	84.9±6.9	60.9±6.0	96.2±0.1	90.1±2.5	98.1±0.3	51.2±0.1
GCN-AN [15]	88.8±5.4	65.6±4.7	95.6±0.3	95.4±1.8	**98.8**±0.3	-
GCN-N [15]	62.6±9.9	56.0±4.4	76.5±4.2	55.1±11	56.9±5.1	-
GRAND [40]	81.3±8.1	56.4±4.5	89.6±1.2	88.7±2.1	91.7±5.5	50.7±1.3
GraFN [41]	58.2±5.8	56.3±3.2	81.4±0.9	97.4±1.5	96.1±3.9	57.3±1.4
ReNode [42]	68.4±6.8	55.1±2.0	82.3±1.5	84.9±3.4	96.3±9.2	50.2±3.1
GFCN	**93.9**±2.3	**68.3**±1.1	**96.3**±0.1	**97.6**±0.5	**98.8**±0.4	**60.3**±0.1

Table 3: Test AUC (%) averaged over 10 runs when 5% of instances are labeled. We also report the standard deviation. Boldface numbers indicate the best anomaly detection performance.

	Dataset					
Method	Cora	Citeseer	Pubmed	Photo	Computers	ogbn-arxiv
OC-SVM [4]	50.2±0.1	50.7±0.5	71.0±1.1	51.9±0.6	47.2±0.8	-
ImVerde [39]	91.1±3.2	64.5±4.5	94.9±0.5	92.2±1.2	98.6±0.6	-
Deep SVDD [9]	58.3±2.8	56.0±0.8	79.9±3.4	52.3±0.8	47.0±1.4	-
Dominant [14]	52.5±0.9	53.9±0.7	53.0±0.5	38.1±0.5	47.2±1.2	-
Deep SAD [10]	72.4±5.7	61.1±4.2	91.7±1.7	89.8±3.1	92.8±2.6	-
GCN [12]	89.2±7.6	63.9±4.7	96.6±0.1	92.3±1.2	98.3±0.3	53.0±0.1
GCN-AN [15]	91.8±5.4	68.3±3.8	96.2±0.2	97.0±0.7	99.1±0.3	-
GCN-N [15]	67.1±5.8	57.4±3.1	76.1±4.8	56.2±9.6	58.2±5.8	-
GRAND [40]	84.7±9.1	56.8±3.1	87.8±1.4	98.4±4.1	97.8±5.8	60.2±5.3
GraFN [41]	61.3±12.6	56.1±4.3	83.0±6.1	98.2±5.6	96.6±7.9	59.8±6.1
ReNode [42]	69.9±3.3	56.1±1.6	82.7±1.1	85.7±2.1	97.5±2.1	60.1±1.9
GFCN	**92.2**±2.3	**71.3**±1.3	**96.7**±0.1	**98.8**±0.4	**99.3**±0.4	**61.1**±0.4

Table 4: Test AUC (%) averaged over 10 runs when 10% of instances are labeled. We also report the standard deviation. Boldface numbers indicate the best anomaly detection performance.

	Dataset					
Method	Cora	Citeseer	Pubmed	Photo	Computers	ogbn-arxiv
OC-SVM [4]	51.8±1.7	51.1±0.9	73.1±0.8	51.7±0.9	47.4±0.8	-
ImVerde [39]	94.5±2.1	68.5±6.9	95.5±0.4	92.8±1.0	99.1±0.4	-
Deep SVDD [9]	59.8±4.8	56.7±1.7	93.2±1.0	53.1±0.6	47.5±1.0	-
Dominant [14]	52.6±0.9	53.9±0.8	53.1±0.4	38.1±0.7	46.6±1.9	-
Deep SAD [10]	72.9±3.3	62.4±4.4	96.7±0.1	89.5±1.9	93.5±1.9	-
GCN [12]	94.5±4.3	68.6±3.3	96.7±0.1	93.3±0.8	98.3±0.3	53.9±0.3
GCN-AN [15]	**95.4**±2.7	72.9±4.3	96.6±0.1	97.9±0.3	99.1±0.3	-
GCN-N [15]	72.3±7.0	60.1±2.1	73.4±5.7	53.6±3.9	58.5±4.6	-
GRAND [40]	86.5±1.3	57.8±6.0	88.8±1.1	94.6±0.7	93.1±0.2	59.9±0.5
GraFN [41]	78.7±7.4	60.3±8.1	83.1±6.3	98.7±4.0	97.2±1.7	62.6±3.3
ReNode [42]	68.3±4.2	54.6±0.8	83.4±1.4	90.4±2.3	99.0±0.3	61.2±1.8
GFCN	94.7±1.0	**76.5**±1.0	**97.3**±0.1	**99.4**±0.2	**99.4**±0.4	**63.0**±0.8

(a) Cora   (b) Citeseer

(c) Pubmed   (d) Photo

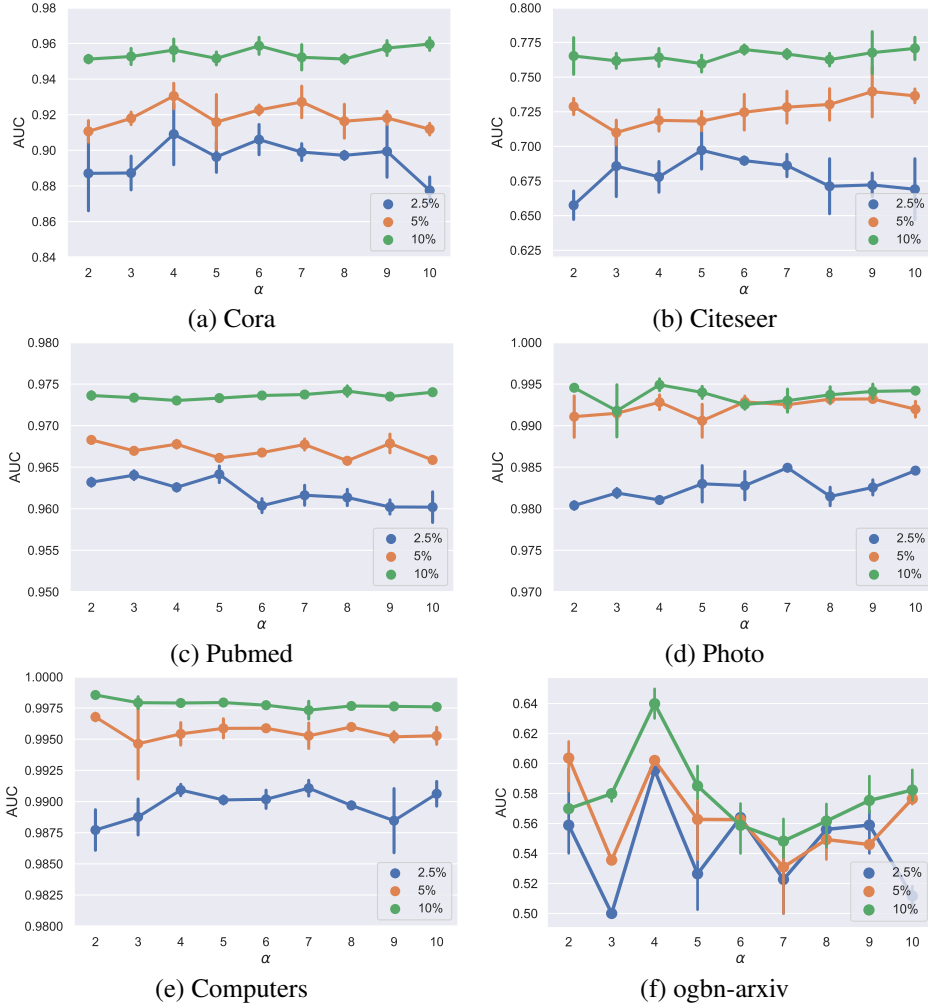(e) Computers   (f) ogbn-arxiv

Figure 4: Effect of weight hyperparameter α on anomaly detection performance (AUC).

We also analyzed the effects of the number of layers and latent representation dimension on the performance of our model using the Cora, Citeseer, and Pubmed datasets (10% of the instances as labeled), and the average AUC results are displayed in Figure 6. As shown in Figure 6 (left), the performance of GFCN remains relatively stable as we increase the depth of the network from 2 to 10 layers. Figure 6 (right) shows the AUC results with the latent representation dimension varying from 10 to 256. As can be seen, our model typically benefits from larger latent dimensions, achieving a good performance with a latent representation dimension equal to 128 for all datasets.

5.7 Visualization

The feature embeddings learned by GFCN can be visualized using the Uniform Manifold Approximation and Projection (UMAP) algorithm [45], which is a dimensionality reduction technique that is particularly well-suited for embedding high-dimensional data into a two- or three-dimensional space. Figure 7 displays the UMAP embeddings of GFCN (left) and GCN (right) using the Cora dataset. As can be seen, the GFCN embeddings are more separable than the GCN ones. With GCN

features, the normal and anomalous instances are not discriminated very well, while with GFCN features these data instances are discriminated much better, indicating that GFCN learns better node representations for anomaly detection tasks.

5.8 Ablation Studies

To validate the influence of our proposed components, we conduct additional experiments for ablation studies by removing components individually. The details and results of our experiments are shown in Table 5, where we report both the average and standard deviation of AUCs over 10 runs on the Cora, Citeseer, and Pubmed datasets. As can be seen, the removal of each component leads to a deterioration in performance. By adding skip connections, an improvement in performance can be observed on all datasets. This indicates that reusing the initial node features in each layer and memorizing information across distant nodes yield improved results. Consistent with prior work, skip connections have been shown to mitigate the vanishing gradient problem in deep neural networks, as well as to improve the accuracy and convergence speed of the network. This has been demonstrated in a number of previous studies, including the
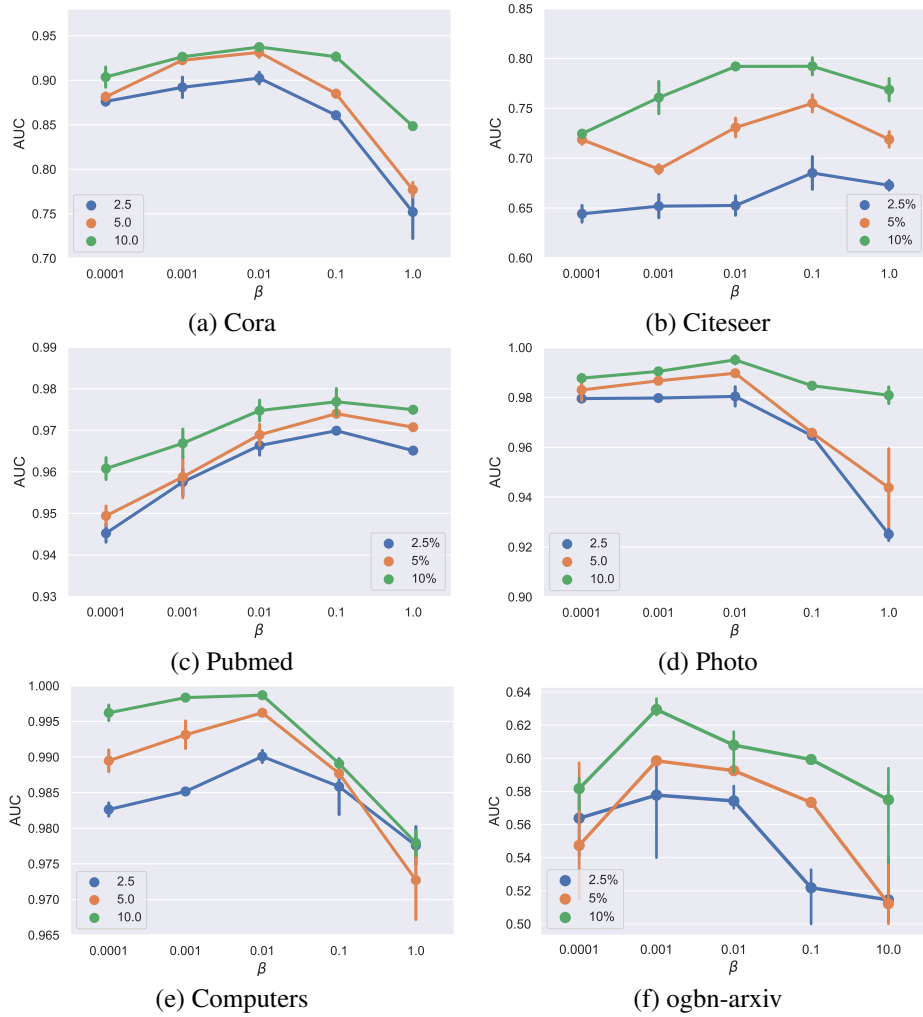
Figure 5: Effect of regularization hyperparameter β on anomaly detection performance (AUC).
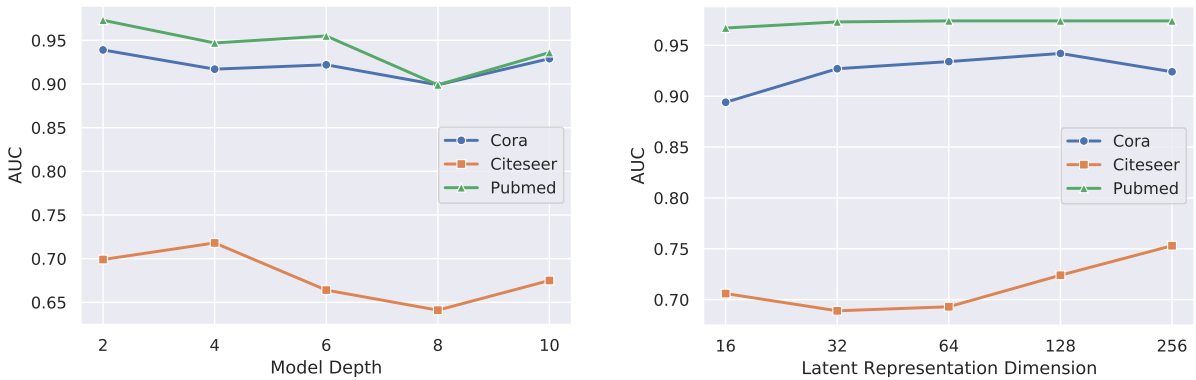

Figure 6: Effects of number of layers (left) and latent representation dimension (right) on anomaly detection performance of our GFCN model using the Cora, Citeseer, Pubmed dataset when 10% of instances are labeled. The AUC results are averaged over 10 runs.

original convolutional neural network with deep residual learning (ResNet) [46] and subsequent work on graph representation learning such as jumping knowledge network (JK-Net) [23], residual graph convolutional network (ResGCN) [26], and graph convolutional networks with initial residual and identity map-

ping (GCNII) [25]. Also, Xu *et al.* [47] examined the optimization dynamics of graph neural networks (GNNs) during their training process, and showed theoretically that skip connections implicitly accelerate the training of GNNs.

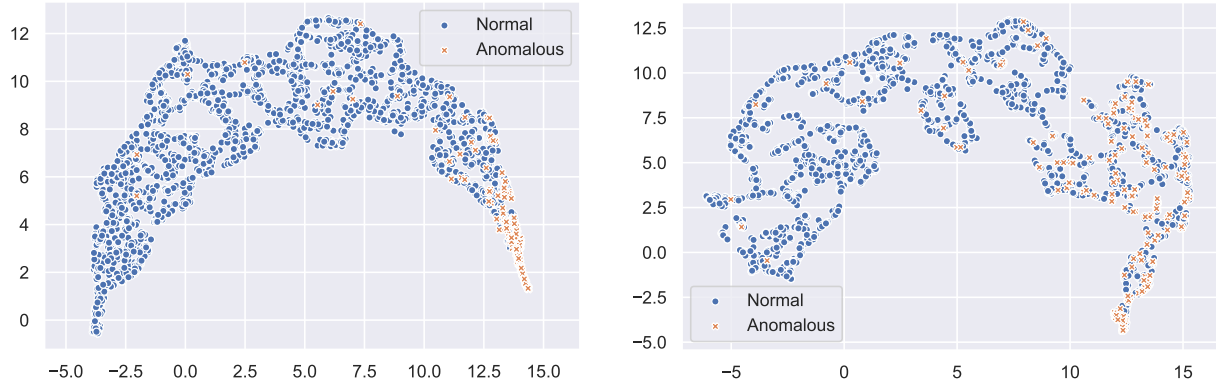As reported in Table 5, the importance of the regularization

Figure 7: UMAP embeddings of GFCN (left) and GCN (right) using the Cora dataset.

term becomes apparent when removing it from the proposed loss function, resulting in a significant drop in AUC. Regularization is important to avoid model overfitting by imposing a penalty to learnable weights. Using regularization term in our loss function yields 9.2%, 8.7%, and 2.9% performance gains over its counterpart model without regularization on the Cora, Citeseer, and Pubmed datasets, respectively.

On the other hand, it is worth pointing out that removing the skip connection and regularization components yields relatively high AUC standard deviations compared to the proposed GFCN model, indicating the robustness of our model.

Table 5: Test AUC (%) averaged over 10 runs when 5% of instances are labeled. We also report the standard deviation.

Method	Cora	Citeseer	Pubmed
Without skip connection	91.2±2.7	64.6±3.8	94.2±0.2
Without regularization	84.7±6.2	59.6±4.1	93.4±0.1
GFCN	**93.9**±2.3	**68.3**±1.1	**96.3**±0.1

6 Conclusion

In this paper, we introduced a graph convolutional network with skip connection for semi-supervised anomaly detection on graph-structured data by learning effective node representations in an end-to-end fashion. The update rule of the proposed graph fairing convolutional network (GFCN) is theoretically motivated by implicit fairing and derived directly from the Jacobi iterative method. GFCN integrates skip connections between the initial feature matrix and each hidden layer. This allows our model to retain and reuse the original node features throughout the network, enabling better information propagation. We also showed that GFCN has the same time and memory complexity as the standard GCN, despite the inclusion of skip connections for improved node representations. Through extensive experiments, we demonstrated the competitive or superior performance of our model in comparison with the current state of the art on five benchmark datasets.While GFCN's intuitive design provides a solid theoretical foundation, it may face

scalability challenges when dealing with very large graphs like many GCN-based methods. For future work, we plan to apply our approach to other downstream tasks on graph-structured data. We also intend to incorporate higher-order neighborhood information into the graph structure of the model, where nodes not only receive latent representations from their 1-hop neighbors, but also from multi-hop neighbors.

Acknowledgments

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[2] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–38, 2021.

[3] K. Doshi and Y. Yilmaz, "Online anomaly detection in surveillance videos with asymptotic bound on false alarm rates," *Pattern Recognition*, vol. 114, 2021.

[4] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, , and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[5] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[6] F. Zhang, S. Kan, D. Zhang, Y. Cen, L. Zhang, and V. Mladenovic, "A graph model-based multiscale feature fitting method for unsupervised anomaly detection," *Pattern Recognition*, vol. 138, 2023.

[7] L. A. S. Arias, C. W. Oosterlee, and P. Cirillo, "AIDA: Analytic isolation and distance-based anomaly detection algorithm," *Pattern Recognition*, vol. 141, 2023.

[8] R. Wang, K. Nie, T. Wang, Y. Yang, and B. Long, "Deep learning for anomaly detection," in *Proc. International Conference on Web Search and Data Mining*, pp. 894–896, 2020.

[9] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proc. International Conference on Machine Learning*, pp. 4393–4402, 2018.

[10] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," in *International Conference on Learning Representations*, 2019.

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[12] T. Kipf and M. Welling, "Semi supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, pp. 1–14, 2017.

[13] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[14] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proc. SIAM International Conference on Data Mining*, pp. 594–602, 2019.

[15] A. Kumagai, T. Iwata, and Y. Fujiwara, "Semi-supervised anomaly detection on attributed graphs," in *Proc. International Joint Conference on Neural Networks*, 2021.

[16] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proc. SIGGRAPH*, pp. 317–324, 1999.

[17] H. Cevikalp, B. Uzun, Y. Salk, H. Saribas, and O. Köpüklü, "From anomaly detection to open set recognition: Bridging the gap," *Pattern Recognition*, vol. 138, 2023.

[18] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *International Conference on Learning Representations*, 2017.

[19] T. Schlegl, P. Seeböck, S. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," *Medical Image Analysis*, vol. 54, pp. 30–44, 2019.

[20] F. D. Mattia, P. Galeone, M. D. Simoni, and E. Ghelfi, "A survey on GANs for anomaly detection," *arXiv preprint arXiv:1906.11632*, 2019.

[21] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?," in *International Conference on Learning Representations*, 2019.

[22] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[23] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. International Conference on Machine Learning*, 2018.

[24] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Combining neural networks with personalized pagerank for classification on graphs," in *Proc. International Conference on Learning Representations*, 2019.

[25] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. International Conference on Machine Learning*, pp. 1725–1735, 2020.

[26] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?," in *Proc. IEEE International Conference on Computer Vision*, pp. 9267–9276, 2019.

[27] Y. Zhang and A. B. Hamza, "Vertex-based diffusion for 3-D mesh denoising," *IEEE Transactions on Image Processing*, vol. 16, pp. 1036–1045, 2007.

[28] E. E. Abdallah, A. B. Hamza, and P. Bhattacharya, "Spectral graph-theoretic approach to 3D mesh watermarking," in *Proc. Graphics Interface*, pp. 327–334, 2007.

[29] E. E. Abdallah, A. B. Hamza, and P. Bhattacharya, "Watermarking 3D models using spectral mesh compression," *Signal, Image and Video Processing*, vol. 3, pp. 375–389, 2009.

[30] G. Taubin, "A signal processing approach to fair surface design," in *Proc. SIGGRAPH*, pp. 351–358, 1995.

[31] G. Taubin, T. Zhang, and G. Golub, "Optimal surface smoothing as filter design," in *Proc. European Conference on Computer Vision*, 1996.

[32] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[33] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[34] F. M. Bianchi, D. Grattarola, C. Alippi, and L. Livi, "Graph neural networks with convolutional ARMA filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3496–3507, 2021.

[35] A. Wijesinghe and Q. Wang, "DFNets: Spectral CNNs for graphs with feedback-looped filters," in *Advances in Neural Information Processing Systems*, 2019.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[37] P. Sen, G. Namata, M. Bilgic, L. Getoor, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[38] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Proc. Relational Representation Learning Workshop, NeurIPS*, 2018.

[39] J. Wu, J. He, and Y. Liu, "ImVerde: Vertex-diminished random walk for learning network representation from imbalanced data," in *Proc. IEEE International Conference on Big Data*, pp. 871–880, 2018.

[40] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Advances in Neural Information Processing Systems*, 2020.

[41] J. Lee, Y. Oh, Y. In, N. Lee, D. Hyun, and C. Park, "GraFN: Semi-supervised node classification on graph with few labels via non-parametric distribution assignment," in *Proc. SIGIR Conference on Research and Development in Information Retrieval*, 2022.

[42] D. Chen, Y. Lin, G. Zhao, X. Ren, P. Li, J. Zhou, and X. Sun, "Topology-imbalance learning for semi-supervised node classification," in *Advances in Neural Information Processing Systems*, 2021.

[43] D. Pickup, X. Sun, P. L. Rosin, R. R. Martin, Z. Cheng, Z. Lian, M. Aono, A. B. Hamza, A. Bronstein, M. Bronstein, S. Bu, U. Castellani, S. Cheng, V. Garro, A. Giachetti, A. Godil, L. Isaia, J. Han, H. Johan, L. Lai, B. Li, C. Li, H. Li, R. Litman, X. Liu, Z. Liu, Y. Lu, L. Sun, G. Tam, A. Tatsuma, and J. Ye, "Shape retrieval of non-rigid 3D human models," *International Journal of Computer Vision*, vol. 120, pp. 169–193, 2016.

[44] S. Biasotti, A. Cerri, M. Aono, A. B. Hamza, V. Garro, A. Giachetti, D. Giorgi, A. Godil, C. Li, C. Sanada, M. Spagnuolo, A. Tatsuma, and S. Velasco-Forero, "Shape retrieval of non-rigid 3D human models," *The Visual Computer*, vol. 32, pp. 217–241, 2016.

[45] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *The Journal of Open Source Software*, 2018.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[47] K. Xu, M. Zhang, S. Jegelka, and K. Kawaguchi, "Optimization of graph neural networks: Implicit acceleration by skip connections and more depth," in *Proc. International Conference on Machine Learning*, 2021.