

Learning Graph Neural Networks with Positive and Unlabeled Nodes

MAN WU, Florida Atlantic University, USA

SHIRUI PAN, Monash University, Australia

LAN DU, Monash University, Australia

XINGQUAN ZHU, Florida Atlantic University, USA

Graph neural networks (GNNs) are important tools for transductive learning tasks, such as node classification in graphs, due to their expressive power in capturing complex interdependency between nodes. To enable graph neural network learning, existing works typically assume that labeled nodes, from two or multiple classes, are provided, so that a discriminative classifier can be learned from the labeled data. In reality, this assumption might be too restrictive for applications, as users may only provide labels of interest in a single class for a small number of nodes. In addition, most GNN models only aggregate information from short distances (e.g., 1-hop neighbors) in each round, and fail to capture *long distance relationship* in graphs. In this paper, we propose a novel graph neural network framework, long-short distance aggregation networks (LSDAN), to overcome these limitations. By generating multiple graphs at different distance levels, based on the adjacency matrix, we develop a long-short distance attention model to model these graphs. The direct neighbors are captured via a short-distance attention mechanism, and neighbors with long distance are captured by a long distance attention mechanism. Two novel risk estimators are further employed to aggregate long-short-distance networks, for PU learning and the loss is back-propagated for model learning. Experimental results on real-world datasets demonstrate the effectiveness of our algorithm.

CCS Concepts: • **Information systems** → *Social networks*.

Additional Key Words and Phrases: Positive unlabeled graph learning, Graph neural networks, Attention

ACM Reference Format:

Man Wu, Shirui Pan, Lan Du, and Xingquan Zhu. 2021. Learning Graph Neural Networks with Positive and Unlabeled Nodes. *J. ACM* 37, 4, Article 111 (August 2021), 25 pages. <https://doi.org/10.1145/3450316>

1 INTRODUCTION

With the rapid development of networking platforms and data intensive applications, graphs are becoming convenient and fundamental tools to model the complex inter-dependence among big scale data. As a result, networks (or graphs) are being widely used in many applications, including citation networks [15], social media networks [29], webpage networks [3], protein-protein interaction networks [7] and so forth. Graph data, however, is inherently sparse and highly complex, making it difficult to carry out graph analytic tasks. For example, graph node classification attempts to categorize nodes in a network into a number of groups, where the essential challenge is the integration of both the graph structure and the node content information.

Authors' addresses: Man Wu, Florida Atlantic University, USA, mwu2019@fau.edu; Shirui Pan, Monash University, Australia, shirui.pan@monash.edu; Lan Du, Monash University, Australia, lan.du@monash.edu; Xingquan Zhu, Florida Atlantic University, USA, xzhu3@fau.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.

0004-5411/2021/8-ART111 \$15.00

<https://doi.org/10.1145/3450316>

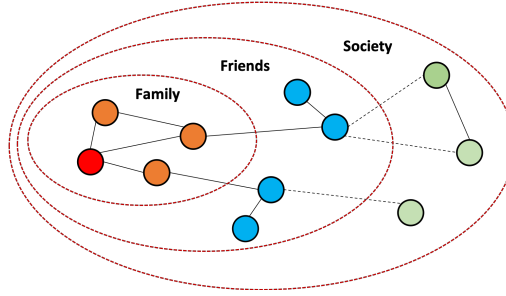


Fig. 1. A conceptual view of short and long distance relationship. Family members are considered short distance relationships and society members are considered long distance relationships.

In order to capture node content and graph structure, many approaches have been proposed recently to embed both structure and node content information of graphs into a compact and low dimensional space for a new representation learning. These existing methods can be roughly categorized into two groups: (1) two-step graph embedding based classification algorithms, and (2) end-to-end graph convolutional neural nets methods.

For two-step graph embedding methods, graph embedding based algorithms first embed nodes in a given graph into vector representation by preserving both structure, node content, and other side information. Then a classical supervised learning algorithm, such as support vector machine, is built from the vector data for classification. Graph embedding algorithms are often learned in an unsupervised manner. They either capture the walk-based similarity between nodes, such as DeepWalk [27], LINE [33], node2vec [9], or apply autoencoder-based models to reconstruct the graph structure information, such as DNGR [2] and SDNE [36]. While being relatively simple, one limitation of these models is that they separate the embedding and the classification task into two steps. As a result, the learned node features may not have best representation for the succeeding classifiers to learn an effective discriminative model for the node classification task.

On the other hand, graph neural network approaches, such as graph convolutional networks (GCNs) [15], employ an end-to-end framework to overcome the limitation of two-step approaches and have achieved impressive performances in the node classification task. The essential idea of GCNs is to generate a *convolutional layer* to exploit the irregular graph structure information and utilize a classification loss function to assist an attributed graph accomplish the classification task. The *graph convolution* operation is described as a filtering process aggregating features from neighboring nodes, *i.e.*,

$$h_i^{(l+1)} = h_i^{(l)} + \text{Aggregate}(h_j)_{j \in \Gamma_i}, \quad (1)$$

here, $h_i^{(l+1)}$ and $h_i^{(l)}$ are latent feature representations of the node v_i at the $l + 1$ and l -th layer, respectively, and $\text{Aggregate}(\cdot)$ is an operation that assorts information from neighbor nodes (Γ_i) of vertex v_i .

After obtaining the new information, GCN applies a neural network to learn a new representation via $o_i^{(l+1)} = W h_i^{(l+1)}$ through a learnable weight matrix W . More specifically, GCN and GraphSage [10] define the $\text{Aggregate}(\cdot)$ as the average, *i.e.*, $\text{Ave}(h_j)_{j \in \Gamma_i}$, or summarization of neighboring feature information that equally considers the significance of each neighbor in the learning process. The recent proposed graph attention network (GAT) aims to learn the weights of different neighbors for aggregating information [34].

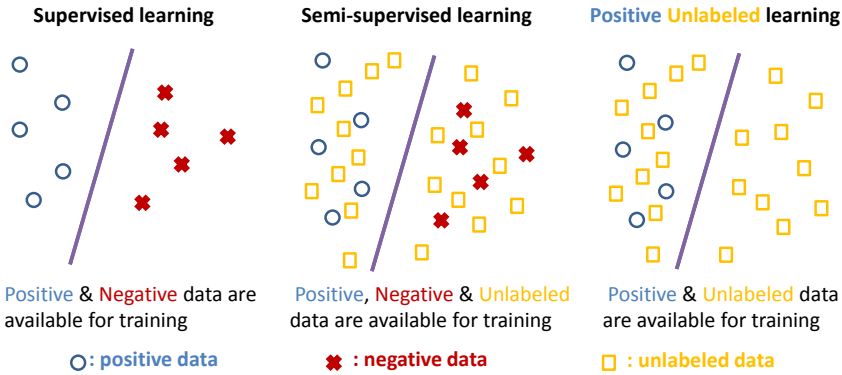


Fig. 2. An example of supervised learning, semi-supervised learning, and positive unlabeled learning. Supervised learning and semi-supervised learning have two or more types (classes) of labeled instances, whereas positive unlabeled learning only has one type of labeled instances (*i.e.* positive samples).

1.1 Motivation

Although commonly used, one of the key limitations of graph neural networks (including GAT [34]) is that they rely on direct (1-hop) neighbor nodes to learn weight information. As a result, *long distance relationship* is largely ignored in the representation learning process [38].

In practice, *long distance relationship* is vitally significant. For example, in social networks, an individual is influenced by her/his neighborhood relations at different distance levels, ranging from short distance relationships (*e.g.* families, friends), to long distance relationships (*e.g.* society, nation states). An example of long distance relationship is illustrated in Figure 1. Since every single relationship is generally sparse and biased, long distance relationship should be additionally considered for graph learning to obtain a comprehensive representation of each node collaboratively.

In addition to the long-short distance relationships, another limitation of existing graph neural networks is that they require users to label data from two or more classes to help facilitate the classification task. This is because that most existing graph neural networks, such as GCN and GAT, are supervised learning or semi-supervised learning approaches, where training samples should include labeled positive and negative samples for binary classification tasks or more types of labeled samples for multi-class classification tasks. Such requirement inevitably imposes significant labeling costs, and in some cases, users may only provide labels of interest for a small number of nodes in one specific class. For example, when surfing the Internet, which is an enormous graph, users may only bookmark pages interesting to them and ignore rest of pages. As a result, only positive samples (*i.e.* bookmarked pages) are labeled and all other pages are unlabeled.

The above observations show a positive unlabeled learning problem setting to recommend pages or news of interest to users. An example of the positive unlabeled learning, compared to supervised learning and semi-supervised learning, is illustrated in Figure 2. As graphs are becoming increasingly popular in applications, many methods are relying on graph neural networks and graph attention mechanisms for learning and analysis. Although positive and unlabeled learning have been previously studied for generic data [20] and graphs [37], this problem has not been addressed and explored by existing graph neural networks.

Motivated by the above observation, our research intends to leverage long-short distance relationships and design new graph neural network approaches for positive and unlabeled graph learning.

1.2 Challenges and Contribution

In this paper, we explore the positive unlabeled graph neural network learning, in which only partial positive nodes are labeled. Considering the extensive usage of graph neural networks as learning frameworks in previous study[10, 15], we summarize following two main challenges:

- *Challenge 1*: How to capture graph structure information from long-distance neighbors? Typically, existing graph neural networks only utilize short-distance information in a single layer.
- *Challenge 2*: How to design an end to end framework for positive unlabeled graph learning? Existing graph neural networks all require labeled nodes from two or more classes to learn a model.

In order to address the above challenges, we propose a novel long-short distance aggregation network (LSDAN) for positive unlabeled (PU) graph learning. For *Challenge 1*, we first generate multiple graphs in different hops based on the adjacency matrix, then develop a long-short distance attention model for these graphs. The long-short distance attention model employs a short-distance attention mechanism to capture the importance of each neighbor node to a target node, and utilizes a long-distance attention approach to model the weights of the different graph with different neighbor nodes for representation learning. For *Challenge 2*, we employ two *novel risk estimators* for positive unlabeled learning and the expected loss is back-propagated for model learning. Experimental results on three real datasets validate the design and effectiveness of our approach. Our contributions can be summarized below:

- We first study *positive unlabeled graph learning* for network node classification task (*i.e.* network transductive learning), and present a new deep learning model LSDAN as a solution.
- We propose a novel attention network for graph data, which captures node significance in both short-distance and long-distance graphs, to model the long-short distance neighboring information in a single layer.
- Experiments on benchmark graph datasets demonstrate that our graph neural network approach outperforms the baseline methods.

The remainder of the paper is structured as follows. Section 2 reviews the related work. Section 3 provides the problem statement. Section 4 presents the proposed algorithm, long-short distance aggregation networks, for PU graph learning. Section 5 illustrates the experimental study, and we conclude this paper in Section 7.

2 RELATED WORK

This work is closely related to graph neural networks, positive unlabeled learning and PU learning for graph data, which are briefly reviewed below.

2.1 Graph Neural Networks

Network node representation aims to map nodes with higher proximities in a network closer to each other in the low-dimensional latent space, which is based on network topology structure only or with side information. For topology structure only embedding methods, most of existing works focused on preserving network structures and properties in embedding vectors [27] [33] [9]. LINE [33] and SDNE [36] seek to preserve the first-order and second-order proximities between nodes based on the first-order and second-order neighbors. DeepWalk [27] employs the random walk sampling strategy to generate the neighborhood of each node. Then, some deep learning approaches [2, 32] have been employed to learn more similar feature representations for nodes which can more easily reach each other within K steps. Aside from topology structure only methods, many approaches are proposed to incorporate side information such as node features [25] [44] [50].

Recently, graph neural networks, which are designed to use deep learning architectures on graph-structured data, have drawn significant attention from the research community. Many solutions are proposed to generalize well-established neural network models that work on regular grid structure to deal with graphs with arbitrary structures [24, 35, 41]. Bruna et al. [1] generalized the convolution operation in the Fourier domain by computing the eigendecomposition of the graph Laplacian. Then, a parameterization of the spectral filters with smooth coefficients was proposed to make them spatially localized [13]. Duvenaud et al. [6] also considered in the form of spectral analysis, and these networks allowed end-to-end learning of prediction pipelines whose inputs were graphs of arbitrary size and shape. Defferrard et al. [5] proposed to approximate the filters by means of a Chebyshev expansion of the graph Laplacian. Finally, Kipf and Welling [15] simplified the previous method by restricting the filters to operate in a 1-hop neighborhood around each node, which can render the extension of CNN to irregular graphs to learn local and stationary features on graphs. Li et al. [21] studied feature learning techniques for graph-structured inputs, they modified Graph Neural Networks [30] to use gated recurrent units and modern optimization techniques and then extended to output sequences. Recently, Hamilton et al. [11] introduced GraphSAGE, a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. Li et al. [19] proposed a generalized and flexible graph CNN taking data of arbitrary graph structure as input. In that way, a task-driven adaptive graph was learned for each graph data while training. You et al. [46] proposed Graph Convolutional Policy Network (GCPN), a general graph convolutional network based model for goal-directed graph generation through reinforcement learning. With the widespread application of attention mechanisms, the development of graph attention network methods in graphs has also been promoted. Velickovic et al. [34] introduced the attention mechanism to graph neural network through specifying different weights to different nodes in a neighborhood. Zhang et al. [49] proposed Gated Attention Networks (GaAN), for learning on graphs. Unlike the traditional multi-head attention mechanism, which equally consumed all attention heads, GaAN used a convolutional sub-network to control each attention head's importance. Graph neural networks has also been used for cross domain text classification [40] or purely unsupervised cross network node classification [39]. Xu et al. [43] study the expressiveness of graph neural networks in terms of their ability to distinguish any two graphs and introduce Graph Isomorphism Network, which is proved to be as powerful as the Weisfeiler-Lehman test for graph isomorphism. You et al. [47] release GraphGym, a powerful platform for exploring different GNN designs and tasks. Chen et al. [4] propose an end-to-end graph learning framework, namely Iterative Deep Graph Learning (IDGL), for jointly and iteratively learning graph structure and graph embedding.

In a recent graph U-Nets design, a gPool [8] procedure is proposed to select top- k nodes to form an induced sub-graph for the next input layer. Although their up-pooling process is efficient, gPool might lose the completeness of the graph structure information, because it only selects top- k nodes, and result in isolated sub-graphs, which hampers the message passing process in subsequent layers. In order to model long-distance relationships, GTNs [48] consider all possible meta-paths within a length limit. Instead of using pooling approaches, we develop a long-short distance attention mechanism to model these graphs by generating multiple graphs at different distances based on the adjacency matrix. The long-short distance attention mechanism serves similar purposes as GTNs [48], but without compromise the graph completeness.

All existing graph neural networks require users to label data from two or more classes to facilitate the classification task. To solve this issue, this work aims to propose a novel long-short distance aggregation network (LSDAN) for positive unlabeled (PU) learning from graphs.

2.2 Positive Unlabeled Learning

Positive unlabeled (PU) learning learns a binary classifier model from positive (P) and unlabeled (U) data. Existing PU methods can be divided into two categories based on how unlabeled data U data are handled. The first category is referred as the two-step strategy, which first identifies possible negative (N) data in U , and then performs the ordinary supervised (PN) learning from both positive and reliable negative examples [20]. The second category is referred to as a direct learning method, and regards U data as N data with smaller weights, which learns classification models from the P and U data directly such as One-class SVM [31], Biased-SVM [22]. However, the former heavily relies on the heuristics in identifying N data, and the latter heavily relies on different choices of the weights of U data, which is computationally expensive to tune.

To deal with this issue, some unbiased PU learning methods [18, 23, 28] are proposed. The main solution is to adopt some novel risk estimators to avoid the bias for PU classification. Specifically, In Niu and Sugiyama [23]'s work, an unbiased risk estimator is proposed to avoid the intrinsic bias for unbiased PU learning. Recently, a non-negative risk estimator [18] is proposed for PU learning, and it is more robust against overfitting when getting minimized, and thus some flexible models can be used given a limited number of P (positive) data. The methods in [18, 23] employ different estimators for positive and unlabeled data, but they mainly focus on the non-graph data or non-relational data. Different from the earlier papers, we focus on the positive unlabeled problem on graph data, by taking both node features and relationships between them into consideration. Moreover, although the methods in [18, 23] tried to tackle the positive unlabeled learning problem, they are limited to feature extraction but cannot be employed for graph feature learning. In our paper, we mainly focus on the graph data and model the long-short distance neighboring information for each node to obtain node features. Through a novel attention network, our method captures node significance in both short-distance and long-distance graphs, which further enhance the feature learning and positive unlabeled graph learning.

2.3 PU Learning for Graph Data

A handful of works have studied PU Learning for graph data, but under different problem settings. Zhao et al. [51] proposed an integrated approach to select discriminative features for graph classification based upon positive and unlabeled graphs. Wu et al. [37] proposed a learning framework for classifying a bag of multiple graphs. They assume each object is represented as a bag of graphs and only partial of bags are positively labeled. So their task is to predict the class label for a whole graph or a bag of graphs.

Our problem setting and learning framework are fundamentally different from these works in three aspects: 1) Existing PU learning for graph data deal with a graph dataset consisting of many graphs, and the task is to predict the class label for a whole graph or a bag of graphs (*i.e.* inductive graph learning). Our goal is to deal with a single large graph and classify nodes in the given graph (*i.e.* transductive graph learning); 2) Existing PU learning on graph are all *shallow* and *biased* models, our algorithm, in comparison, is an *unbiased* and *deep* neural network model; and 3) In order to achieve PU learning for graph node classification, we propose to combine graph feature learning into the classification task, using specifically designed objective function.

3 PROBLEM STATEMENT

Graph: A graph is represented as $G = (V, E, X, Y)$, where $V = \{v_i\}_{i=1, \dots, N}$ is a vertex set representing nodes in a graph, and $e_{i,j} = (v_i, v_j) \in E$ is an edge indicating relationships between nodes. The topological structure of graph G can be represented by an adjacency matrix A , where $A_{i,j} = 1$ if $(v_i, v_j) \in E$; otherwise $A_{i,j} = 0$. $\mathbf{x}_i \in X$ indicates content features associated with each

Table 1. Summary of Notations and Symbols.

Notations	Descriptions
$G = (V, E, X, Y)$	An attributed graph
V, E	Node set and edge set of G
P, U	Labeled and unlabeled node set of G
$n, V = n$	Number of nodes in G
$A \in \mathbb{R}^{n \times n}, A^k \in \mathbb{R}^{n \times n}$	Adjacency matrix (A) and length- k walk matrix (A^k) of G
$X \in \mathbb{R}^{n \times m}$	Feature matrix of G
$\mathbf{x}_i \in \mathbb{R}^m$	Feature vector of node v_i . Each node has m dimensional features
$y_i \in \mathbb{R}^{\{1,u\}}$	Label of node v_i . A node is either positive (1) or unlabeled (u).
$Y \in \mathbb{R}^{n \times 2}$	Label matrix of G
κ	the maximum hops for long-short distance aggregation
$\sigma(\cdot)$	a non-linear activation function
$\mathbf{W}^{(1)} \in \mathbb{R}^{d \times m}$	A shared weight matrix for short-distance feature aggregation learning
$\alpha_{i,j}$	Weight value of the neighbor v_j for node v_i
$a_{st}(\cdot)$	Short-distance attention function
$\mathbf{H} \in \mathbb{R}^{n \times d}$	Short-distance attention feature embedding results
$\mathbf{O} \in \mathbb{R}^{n \times m}$	Long-short distance attention feature embedding results
$t_{i,j}$	Attention coefficient between v_i and v_j
$\mathbf{T} \in \mathbb{R}^{n \times n}$	Attention coefficient matrix consisting of $t_{i,j}$
$B^k \in \mathbb{R}^{n \times n}$	k -hop adjacency matrix derived from A^k
$\mathbf{W}^{(2)} \in \mathbb{R}^{d \times m}$	A shared weight matrix for long-short distance feature attention learning
$a_{ls}(\cdot)$	Long-short distance attention function
c_i	the attention coefficient computed by long-short distance attention function a_{ls}
$R(f)$	the expected loss/risk
$R_p^+(f)$	the expected loss/risk for positive class
$R_n^-(f)$	the expected loss/risk for negative class
$\hat{R}_{pn}(f)$	the empirical loss/risk in traditional binary classification
$\hat{R}_p^+(f)$	the empirical loss/risk for positive class with loss $\mathcal{L}(f(o_i^p), 1)$
$\hat{R}_p^-(f)$	the empirical loss/risk for positive class with loss $\mathcal{L}(f(o_i^p), 0)$
$\hat{R}_n^-(f)$	the empirical loss/risk for negative class
$\hat{R}_{pu}(f)$	the empirical loss/risk for positive and unlabeled learning
$\hat{R}_u^-(f)$	the empirical loss/risk for unlabelled class

node v_i . $y_i \in Y = \{+1, 0\}$ is the ground-truth class label for each node, where “1” denotes positive class, and “0” denotes negative class (not positive). If a node v_i is of interest to a user, then $y_i = 1$, or $y_i = 0$ otherwise. It is worth noting that although the ground-truth label of each node is binary (1 or 0), only a small portion of positive nodes are labeled in PU graph learning, so the labeled set only has positive samples.

Positive Unlabeled Graph Learning (PUGL): . Assume $V = P \cup U$, where P are the labeled nodes ($\forall v_i \in P, y_i = 1$) and U are unlabeled nodes. Given a graph $G = (V, E, X, Y)$, Positive Unlabeled Graph Learning (PUGL) **aims** to learn a binary classifier model, $f : (A, X; P) \mapsto Y$, to predict the class labels for unlabeled nodes U . In this paper, we propose the first deep learning model for PUGL.

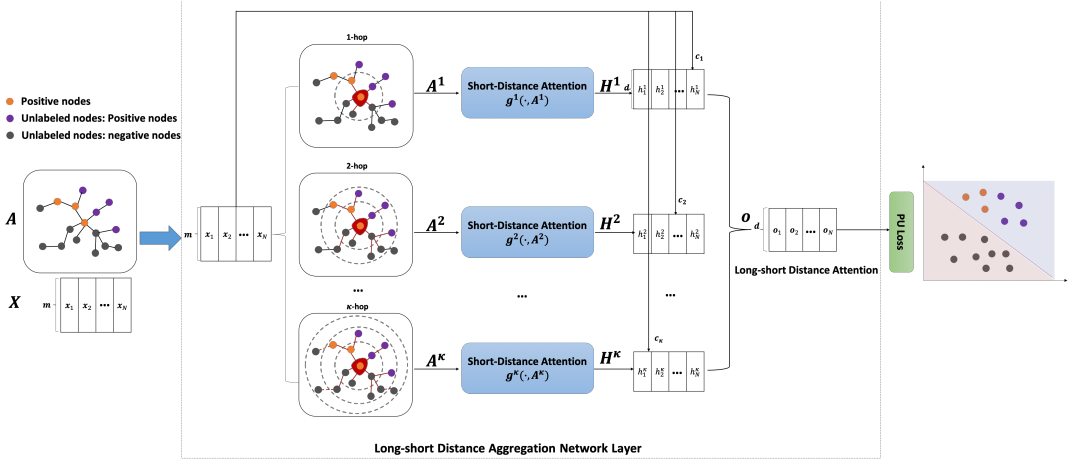


Fig. 3. The overall architecture of the proposed long-short distance aggregation network (LSDAN) model. LSDAN uses higher-order network topology structures and node content (X) to progressively learn a long-short distance attention model, whose outputs are integrated into a learning objective function to achieve optimized PU graph learning outcomes. Here, LSDAN uses higher order adjacency matrices to capture *long distance relationship* w.r.t. a target node.

For ease of understanding, Table 1 summarizes major symbols and notations used in the paper.

4 LONG-SHORT DISTANCE AGGREGATION NETWORKS FOR PU GRAPH LEARNING

In this section, we present our proposed LSDAN algorithm for PU Graph learning. Our learning objectives are to (1) capture the *long-short distance relationship* between nodes, and (2) enable PU learning on a graph. We will first present our long-short distance attention network which exploits both short-distance and long-distance attention for *long-short distance relationship* modeling. Then we present two risk estimators for PU learning. Our framework, as shown in Figure 3, mainly consists of three components:

- **Short-Distance Attention.** For the input X and an adjacent matrix A , a short-distance self attention mechanism is applied to learn a representation for each node.
- **Long-short Distance Attention.** Given an input graph G , we will first generate multi-hop graph representation based on adjacent matrix $A^1, A^2, \dots, A^K$. The matrix A^k captures the neighbors in the k -th hop of the graph G . We develop a long-distance attention approach to automatically determine the weights of different graphs $A^1, A^2, \dots, A^K$.
- **Positive Unlabeled Learning.** Based on our long-short distance attention model, we develop a deep architecture for learning the graph representation of each node. Then the unbiased risk estimator and the non-negative risk estimator are used to estimate the classification loss, respectively. The loss is further back-propagated to the learning progress in an end to end learning framework.

4.1 Short-Distance vs. Long-Distance

Definition 4.1. SHORT-DISTANCE: Short-distance is defined as the distance from direct (1-hop) neighbor nodes to a target node.

The (normalized) adjacency matrix A characterizes the first-order proximity to model the direct relationship (1-hop) between vertices.

Definition 4.2. LONG-DISTANCE: Long-distance is defined as the distances of k -hop neighbors ($k > 1$) to a target node.

In order to capture long-distance relation for each node, we propose to consider k -distance (with varying $k \in [1, \kappa]$) relational information from the network for graph learning. Given an input graph G , we will first generate multi-hop graph representation based on adjacent matrix $A^1, A^2, \dots, A^\kappa$. The matrix A^k captures the neighbors in the k -th hop of the graph G , as shown in Figure 4. Therefore, the k -distance relationship can be captured by:

$$A^k = \underbrace{A \cdot A \cdots A}_k \quad (2)$$

where $A^k_{i,j}$ refers to the k -hop link relation between node v_i and v_j . In other words, if $A^k_{i,j} \neq 0$, it means that node v_i and v_j have a k -hop relation, or zero otherwise.



Fig. 4. A conceptual view of long-shot distance attention mechanism. Short-distance corresponds to the 1-hop neighbors which is captured by the adjacency matrix A . Long-distance is defined as the distances from k -hop neighbors ($k > 1$) to a target node.

4.2 Long-short Distance Attention

In this subsection, we propose to differentiate and combine short-distance attention and long-distance attention to learn network feature representation PU learning.

4.2.1 Short-Distance Attention. Given the input $X \in \mathbb{R}^{n \times m}$ denoting content matrix for all n nodes where each node has m dimension feature, and an adjacent matrix $A \in \mathbb{R}^{n \times n}$, a short-distance self attention mechanism is applied to learn a representation for each node, which aims to capture the node features of the whole graph with short distance by aggregating feature values within each node's neighborhood. Specifically, the input is a set of node features, $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^m$ denotes feature vector for node v_i where m denotes the number of input features of each node. The output of the short-distance attention is a new set of node features, $\mathbf{H} \in \mathbb{R}^{n \times d} = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n\}$, where $\mathbf{h}_i \in \mathbb{R}^d$ and d denotes the number of embedding features of each node.

$$\mathbf{h}_i = g \left(\sum_j \alpha_{i,j} A_{i,j} \mathbf{W}^{(1)} \mathbf{x}_j \right), \quad (3)$$

where g is a non-linear activation function, $A_{i,j}$ serves as a mask to only aggregate node V_i 's direct neighbors (short-distance neighbors) for feature learning. $\alpha_{i,j}$ is weight value capturing the

importance of neighbor v_j for node v_i . To automatically learn the parameter $\alpha_{i,j}$, a short-distance self-attention mechanism is developed.

To compute $\alpha_{i,j}$, a shared linear transformation is applied to each node through multiply a shared weight matrix $W \in \mathbb{R}^{d \times m}$ in the initial step. Then an attention coefficient $t_{i,j}$ is computed by an attention function $a_{tt}(\cdot)$:

$$t_{i,j} = a_{tt}(W^{(1)} \mathbf{x}_i, W^{(1)} \mathbf{x}_j), \quad (4)$$

which measures the importance of vertex j to vertex i . In the most general formulation, the model allows every node to attend on every other node, dropping all structural information. We inject the graph structure into the mechanism by performing masked attention, masking out all other nodes except direct neighbors based on the adjacency matrix A .

Furthermore, in order to make coefficients comparable among vertices, a softmax function is utilized to normalize attention coefficients:

$$\alpha_{i,j} = \text{softmax}_j(t_{i,j}) = \frac{\exp(t_{i,j})}{\sum_j A_{i,j} \exp(t_{i,j})}, \quad (5)$$

In the experiment, the attention mechanism a_{tt} is instantiated with a dot product (parametrized by a weight vector $\mathbf{r} \in \mathbb{R}^{2d}$) and a LeakyReLU [42] nonlinearity. Fully expanded out, the normalized attention coefficients can be expressed as:

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(\mathbf{r}^T [W^{(1)} \mathbf{x}_i \oplus W^{(1)} \mathbf{x}_j]))}{\sum_j A_{i,j} \exp(\text{LeakyReLU}(\mathbf{r}^T [W^{(1)} \mathbf{x}_i \oplus W^{(1)} \mathbf{x}_j]))}, \quad (6)$$

where $a \oplus b$ denotes the concatenation operation of vector a and b .

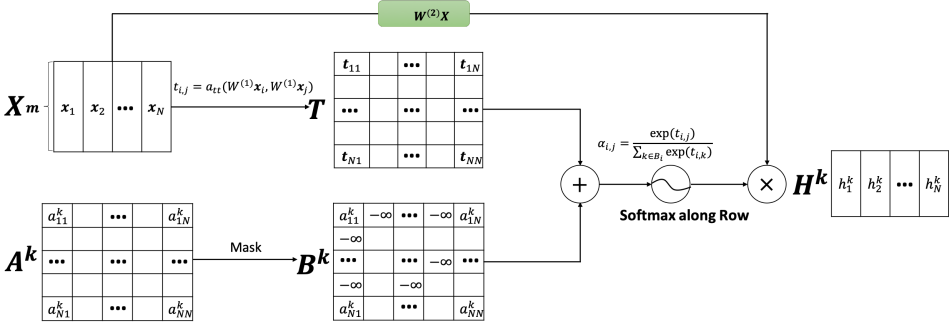


Fig. 5. The mechanism of the long-distance attention, which takes X and A^k as inputs, and computes the importance of nodes via Eq. (4) to produce the attention coefficient T and get a matrix B^k by masking out all the other nodes except neighbors based on the A^k . After that, it will produce a new set of node features H^k .

4.2.2 Long-Distance Attention. To capture long-distance relation between nodes, we need to aggregate embedding from different range of neighborhoods to produce a unified representation. Because neighbors from different distances contribute differently to learning the representation, we propose a *Long-Distance Attention* scheme to capture the significance of each k -hop distance graph neighbors.

Specially, for each A^k , $k \in \{1, \dots, \kappa\}$, we will perform the self attention to learn the embedding H^k for each node (as shown in Figure 5). In order to obtain k -hop neighborhood for each node, we define a matrix B^k to capture whether two nodes (v_i and v_j are k -hop neighbors).

$$B_{i,j}^k = \begin{cases} 1 & \text{if } A_{i,j}^k \neq 0 \\ 0 & \text{Otherwise.} \end{cases} \quad (7)$$

By using B^k to mask out other nodes except k -hop neighbors, we can learn k -hop neighbor aggregated feature embedding as follows.

$$\mathbf{h}_i^k = \mathbf{g} \left(\sum_j \alpha_{ij} B_{ij}^k \mathbf{W}^{(1)} \mathbf{x}_j \right), \quad (8)$$

We then use the original input $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^m$ as the key of the attention mechanism, and perform attention on each graph output $H^k = \{\mathbf{h}_1^k, \mathbf{h}_2^k, \dots, \mathbf{h}_n^k\}$, $\mathbf{h}_i^k \in \mathbb{R}^d$, an attention coefficient c_i^k is computed by an attention function $a_{ls}(): \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$:

$$c_i^k = a_{ls}(\mathbf{h}_i^k, W^{(2)} \mathbf{x}_i), \quad (9)$$

where $W^{(2)}$ is a shared weight matrix for long-short feature attention learning, characterizing the consistency between short-distance and long-distance aggregated features ($W^{(2)}$ also enforces the input $\mathbf{x}_i^k$ of node i to have the same dimension as the k -hop embedding features $\mathbf{h}_i^k$). In this paper, we denote $a_{ls}()$ as a dot-product attention function. After that, we further normalize the weight c_i^k with a softmax layer.

$$c_i^k = \frac{\exp(c_i^k)}{\sum_{k=1}^K \exp(c_i^k)}. \quad (10)$$

After implementing the attention, final embedding output $\mathbf{O} \in \mathbb{R}^{n \times d} = \{\mathbf{o}_1, \dots, \mathbf{o}_n\}$, $\mathbf{o}_i \in \mathbb{R}^d$:

$$\mathbf{o}_i = \sum_{k=1}^K c_i^k \mathbf{h}_i^k. \quad (11)$$

4.3 Deep Long-short Distance Aggregation Networks

The short-distance attention and long-distance attention components are integrated into a unified layer, *Long-short Distance Aggregation Network Layer* (LSDAN), which serves as a building block to construct a deep architecture for node classification in a single network, as shown in Figure 6. The LSDAN layers are stacked in the following way:

- The input U^{l+1} to the $(l+1)$ th layer is the sum of the output O^l and the input U^l from layer l :

$$U^{l+1} = U^l + O^l. \quad (12)$$

- A *residual connection* [12] around two sub-layers and the multi-graph information $A^1, A^2, \dots, A^K$ are used in different layers. The residual connection method provides the input without any transformation to the output of the $(l+1)$ th layer, which makes the $(l+1)$ th layer learn something new about the network.
- At the first layer of the network ($l=1$), let $U^1 = X$, we will not use the residual connection (i.e. $U^2 = O^1$), because we need to first map a high-dimensional node representation to a low-dimensional representation.
- At the last layer of the network ($l=L$), we do not use the residual connection, and map the embedding of nodes to the 2-dimensional representation for PU classification.

By this way, we can build arbitrary deep long-short distance aggregation networks to effectively learn graph representation, by leveraging the long-short distance neighboring information.

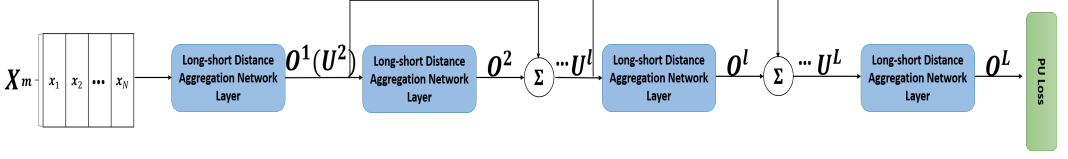


Fig. 6. The architecture of multiple layers of the proposed model.

4.4 Positive and Unlabeled Graph Learning

After integrating the structure and content information via a deep long-short distance aggregation network, we will obtain the new representation $O^L = \{\mathbf{o}_1^L, \dots, \mathbf{o}_N^L\}$, $\mathbf{o}_i^L \in \mathbb{R}^2$ in the final layer. One key question has arisen as *how can we perform positive unlabelled learning from this new representation?*

We first formulate the traditional binary classification problem as a risk minimization problem, and then employ two effective positive unlabeled learning methods to approximate the risk for PUGL. An *unbiased risk estimator* and a *non-negative risk estimator* are used to approximate the risk for PU learning, respectively. By minimizing the risk, our model is learned in an end-to-end manner.

4.4.1 Traditional Binary Classification. Given a set of the obtained representations $O^L = \{\mathbf{o}_1^L, \dots, \mathbf{o}_N^L\}$, $\mathbf{o}_i^L \in \mathbb{R}^2$ where $\mathbf{o}_i$ is new feature representation of node i . In the traditional binary classification, we need to learn a model $f : \mathcal{O} \rightarrow \mathcal{Y}$, to classify each node $\mathbf{o}_i$ into the predefined categories $\mathcal{Y} = \{+1, 0\}$, which is the ground-truth label of the node (1 denotes positive samples, 0 denotes negative samples).

Let $\mathcal{L} : \mathbb{R} \times \{+1, 0\} \rightarrow \mathbb{R}$ be a loss function, then $\mathcal{L}(y', y)$ measures the predicting loss for an output y' when the ground truth is y . Let f be a mapping function, and $f(o)$ maps the input o in the range $(0, 1)$. The traditional binary classification problem is formulated as a risk minimization problem:

$$R(f) = \mathbb{E}[\mathcal{L}(f(O), Y)] = \pi_p R_p^+(f) + \pi_n R_n^-(f), \quad (13)$$

where $R_p^+(f) = \mathbb{E}_p[\mathcal{L}(f(O), +1)]$ and $R_n^-(f) = \mathbb{E}_n[\mathcal{L}(f(O), 0)]$ are the expected loss for positive and negative samples. Here, we denote $\pi_p = p(Y = +1)$ be the *class-prior probability*, $\pi_n = p(Y = 0) = 1 - \pi_p$. π_p is assumed to be known throughout the paper, and it can be estimated from positive data [14].

Therefore, for traditional binary classification problem i.e., positive and negative learning (PN Learning), we can minimize an approximated $R(f)$ by,

$$\hat{R}_{pn}(f) = \pi_p \hat{R}_p^+(f) + \pi_n \hat{R}_n^-(f), \quad (14)$$

where $\hat{R}_p^+(f) = (1/n_p) \sum_{i=1}^{n_p} \mathcal{L}(f(\mathbf{o}_i^p), +1)$ and $\hat{R}_n^-(f) = (1/n_n) \sum_{i=1}^{n_n} \mathcal{L}(f(\mathbf{o}_i^n), 0)$. Here, n_p and n_n denote the number of positive/negative samples, respectively.

4.4.2 Unbiased Risk Estimator for PU Learning. For positive unlabeled (PU) learning, however, negative training data is unavailable. As a result, we need to estimate $\hat{R}_n^-(f)$ via Eq. (14). Here, we use a novel unbiased risk estimator to approximate $\hat{R}_n^-(f)$ [28]. Specifically, the expected loss $\hat{R}_n^-(f)$ of negative samples can be obtained by the expected loss $\hat{R}_p^+(f)$ of positive samples and the expected loss $\hat{R}_u^-(f)$ of unlabeled samples, and is defined as

$$\pi_n \hat{R}_n^-(f) = -\pi_p \hat{R}_p^+(f) + \hat{R}_u^-(f), \quad (15)$$

where $\hat{R}_p^-(f) = (1/n_p) \sum_{i=1}^{n_p} \mathcal{L}(f(o_i^p), 0)$, and $\hat{R}_u^-(f) = (1/n_u) \sum_{i=1}^{n_u} \mathcal{L}(f(o_i^u), 0)$. Here, n_p and n_u denote the number of positive/unlabeled samples, respectively.

Therefore, for positive unlabeled (PU) learning, the risk $R(f)$ can be approximated by,

$$\hat{R}_{pu}(f) = \pi_p \hat{R}_p^+(f) - \pi_p \hat{R}_p^-(f) + \hat{R}_u^-(f). \quad (16)$$

4.4.3 Non-negative Risk Estimator for PU Learning. Although the unbiased risk estimator can efficiently solve the positive unlabeled learning problem. However, Eq. (16) may cause the risk value get negative, because there's a negative sign in front of $\hat{R}_p^-(f)$. The $\hat{R}_p^-(f)$ refers to the sample in the positive sample set which is predicted to get the expected risk value with the negative label through the model, that is: if the prediction is the negative sample, loss is 0; if the prediction is not the negative sample, loss is positive. This will cause overfitting problem for PU Learning. Motivated by Kiryo et al. [17], we employ a non-negative risk estimator $\hat{R}_{pu}(f)$, given as follows,

$$\hat{R}_{pu}(f) = \pi_p \hat{R}_p^+(f) + \max \left\{ 0, \hat{R}_u^-(f) - \pi_p \hat{R}_p^-(f) \right\}. \quad (17)$$

In our paper, we will minimize the unbiased empirical risk and non-negative empirical risk, respectively. Specifically, let f be a mapping function, and we use a Sigmoid activation function $f(o) = \frac{1}{1+\exp(-o)}$ to map the input $\mathbf{o}$, which can be learned by our graph neural network module in Eq. (11), to the range (0,1). The Logistic Loss is used in the paper, and the loss function $\mathcal{L}(y_i', y_i)$ of each sample is defined as:

$$\mathcal{L}(y_i', y_i) = -[y_i * \log(y_i') + (1 - y_i) * \log(1 - y_i')], \quad (18)$$

where y_i' and y_i are the model predict score and the ground-truth for each sample. The expected loss/risk can be computed by the unbiased empirical risk and non-negative empirical risk via Eq. (16) and Eq. (17), respectively.

4.5 Algorithm Description

Our algorithm is illustrated in Algorithm 1. Given a graph $G = (V, E, X, Y)$, the goal of Positive Unlabeled Graph Learning (PUGL) is to learn the node representations and learn a binary classifier model, $f : (A, X; P) \mapsto Y$, to predict class labels for unlabeled nodes in U .

The algorithm first obtains adjacency matrix A and its power matrices A^k from G by Eq. (2) (Step 1). After that, it uses labelled node set P and unlabelled node set U to calculate *class-prior probability* π_p from G (Steps 2-4). Next, the algorithm carry out long-short distance aggregation to effectively learn node representations by leveraging the long-short distance neighboring information. The final output of node representations are denoted by O^L (Steps 5-18). Finally, the algorithm formulates a binary classification task as a risk minimization problem, and employs two effective positive unlabeled learning methods to approximate the risk for PUGL. An unbiased risk estimator and a non-negative risk estimator are used to approximate the risk for PU learning, respectively. By minimizing the risk, the loss can be further back-propagated in our proposed model to guide the representation learning to achieve better PU graph learning results (Steps 19-23).

4.6 Time Complexity Analysis

Given a graph $G = (V, E, X, Y)$, the proposed Long-short Distance Aggregation Network Layer (LSDAN) consists of two parts: κ short-distance attentions, and long-short distance attention, where κ denotes the number of hops. It is worth noting that the calculation of the matrix A^k can be done in advance, and there is no need to recalculate the matrix A^k during the training process. The calculation of A^k requires $O(|V|^3)$ time complexity. The time complexity for computing each short-distance attention is $O(|V|md + |E|d)$, where $|V|$ and $|E|$ are the numbers of nodes and edges

Algorithm 1 LSDAN for Positive Unlabeled Graph Learning

Require:

- (1) Graph: $G = (V, E, X, Y)$
- (2) Maximum hops for long-short distance aggregation: κ
- (3) Maximum graph neural network layers: L

Ensure:

A binary classifier model, $f : (A, X; P) \mapsto Y$.

- 1: $\{A, \dots, A^\kappa\} \leftarrow$ Obtain adjacency matrix A and its power matrices A^k from G
 - 2: $P \leftarrow \{v_i | v_i \in V, y_i = 1\}$. Labelled node set.
 - 3: $U \leftarrow \{V \setminus P\}$. Unlabelled node set.
 - 4: $\{\pi_p\} \leftarrow$ calculate *class-prior probability* from G .
 - 5: **while** not convergence **do**
 - 6: **for** graph neural network layer $l=1$ to L **do**
 - 7: **if** first layer $l == 1$ **then**
 - 8: $\mathbf{V}^1 \leftarrow X$
 - 9: **else if** last layer $l == L$ **then**
 - 10: $\mathbf{V}^L \leftarrow \mathbf{O}^{L-1}$
 - 11: **else**
 - 12: $\mathbf{V}^l \leftarrow \mathbf{V}^{l-1} + \mathbf{O}^{l-1}$
 - 13: **end if**
 - 14: **for** each hop distance $k=1$ to κ **do**
 - 15: $\mathbf{H}^{l,k} \leftarrow$ Learn embedding for each A^k by Eq. (3)
 - 16: **end for**
 - 17: $\mathbf{O}^l \leftarrow$ Learn the new output representation based on the $\mathbf{H}^l$ by Eq. (11)
 - 18: **end for**
 - 19: **if** *Unbiased Positive Learning* **then**
 - 20: $[W^{(1)}, W^{(2)}] \leftarrow$ Back-propagate loss gradient using unbiased empirical risk Eq. (16) and Eq. (18)
 - 21: **else if** *Non-negative Positive Learning* **then**
 - 22: $[W^{(1)}, W^{(2)}] \leftarrow$ Back-propagate loss gradient using non-negative empirical risk Eq. (17) and Eq. (18)
 - 23: **end if**
 - 24: **end while**
-

in the graph, respectively, and m and d denote the dimensions of the input feature and output feature of a single layer, respectively. The time complexity of long-short distance attention is $O(md)$. Therefore, the overall time complexity of the proposed Long-short Distance Aggregation Network Layer (LSDAN) is $O(\kappa|V|^3 + \kappa|V|md + \kappa|E|d + md)$.

In real-world networks, the number of nodes $|V|$ and the number of edges $|E|$ are much larger than feature dimension m , embedding size d , and the the maximum hops for long-short distance aggregation κ , where V and E are more than thousands, and m , d , and κ are in hundreds maximum. In addition, we know that $|E| < |V|^2$. Therefore, LSDAN's complexity is asymptotically bounded by $O(\kappa|V|^3)$.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate our model against state-of-the-art algorithms on three real-world datasets. Furthermore, we also provide detailed experimental analysis to show more insights of our model.

5.1 Experiment Setting

Datasets We employ three widely used citation network datasets (Cora, Citeseer, DBLP) for node classification [26, 45]. The details of the experimental datasets are displayed in Table 2. The Cora dataset contains 2708 nodes, 5429 edges with 7 classes and 1433 features. The Citeseer dataset contains 3312 nodes, 4732 edges with 6 classes and 3703 features. The DBLP dataset contains 5818 nodes, 3633 edges with 4 classes and 1587 features. As these datasets have multiple classes, we select the class with the relatively large number of samples as P (positive) class, and all the other classes are regarded as N (negative) class. Specifically, for the Cora, Citeseer, and DBLP datasets, we select the class with the label is 3, 2, and 1, respectively, as P (positive) class and the remaining classes as N (negative) class. After selecting the positive class, we convert the original classification problems of each dataset into binary classification tasks.

Table 2. Statistics of three datasets.

Dataset	Node	Edges	Classes	Features
Cora	2708	5429	7	1433
Citeseer	3312	4732	6	3703
DBLP	5818	3633	4	1587

Baselines To the best of our knowledge, there is no existing study on positive unlabeled graph neural network learning. To make a fair comparison and evaluate the effectiveness of our design, we select the following baselines with necessary adaption.

We first compare our model with the classical PU learning methods, which focus on the one-step strategy and two-step strategy.

- **OC-SVM:** OC-SVM [31] (One-class SVM algorithm) is a classical machine learning algorithm based on support vector machine. It only uses positive examples from the node content to build a binary classifier. In our experiments, we use node features as the input of One-class SVM.
- **LINE_OC-SVM:** We first use an unsupervised network embedding method (LINE [33]) to learn node representation. After that, the learned features are used as the input of One-class SVM.
- **GAE_OC-SVM:** We use an unsupervised graph embedding method (Graph Auto-Encoders, GAE [16]) to learn node representation by using both the adjacency matrix of nodes and the features of nodes. After that, the learned features are used as the input of One-class SVM.
- **Roc-SVM:** Roc-SVM [20] uses two step strategies to build a classifier from the node content, and combines the Rocchio method and the SVM technique for PU learning algorithm. In our experiments, we use the features of nodes as the input of Roc-SVM.
- **LINE_Roc-SVM:** We use an unsupervised network embedding method (LINE [33]) to learn node representation by only using the adjacency matrix of nodes. After that, the learned features are used as the input of Roc-SVM.
- **GAE_Roc-SVM:** We use an unsupervised graph embedding method (Graph Auto-Encoders, GAE [16]) to learn node representation by using both the adjacency matrix of nodes and the features of nodes. After that, the learned features are used as the input of Roc-SVM.

In addition the above baseline, we also compare our algorithm with different deep learning models. Note that we have integrated the *unbiased risk estimator* and *non-negative risk estimator* into the following models for PU learning.

- **FC:** Full-connected network only applies node features to a multiple layer perceptron (MLP) to learn node representation without using the adjacency matrix of nodes.
- **FS:** Full-connected self-attention network uses the node features with a self-attention network to obtain the representation without the adjacency matrix of nodes.
- **GCN:** GCN uses the graph convolutional network [15] to integrate structure and content information of nodes to learn node representation using the adjacency matrix of nodes.
- **GAT:** GAT uses the graph attention nets [34] to exploit structure and content information of nodes to obtain node representation using the adjacency matrix of nodes. Note that, in the experiments, we only utilize one attention head.
- **GATH:** GATH uses the graph attention nets [34] to exploit structure and content information of nodes to obtain node representation using the adjacency matrix of nodes. In addition, multi-head attention is further utilized to stabilize the learning process and encapsulate detailed information about the neighborhood.

Our method:

- **LSDAN_UPU:** LSDAN_UPU employs a long-short distance aggregation network to exploit structure and content information of nodes to obtain the final graph representation, and the unbiased risk estimator is utilized for PU learning.
- **LSDAN_NNPU:** LSDAN_NNPU employs a long-short distance aggregation network to exploit structure and content information of nodes to obtain the graph representation, and the non-negative risk estimator is utilized for PU learning.

Experimental Setup For fairness of comparison, we randomly split each PN dataset into positive and unlabeled set. Following Kiryo et al. [17], we sample N_{PN} (the total number of positive nodes) nodes from N as negative class. Then we select $p * N_{PN}$ nodes from P as the training set, the rest positive nodes and negative nodes are used as the unlabeled set (p is the percentage of training (positive) nodes). We conduct 10 trials of randomly splitting, and report the average **F1 score** as final experimental results.

All models were implemented in TensorFlow with the Adam optimizer with a learning rate of $1e^{-4}$ for 500 steps. For parameter setting, we set the embedding dimension of nodes to 64 for all methods. We choose 2 layers for GCN-PU and GAT-PU, where the first GCN/GAT layer contains 64 hidden units, and the second layer contains 2 hidden units for classification. For the proposed LSDAN, the number hops κ is set to 4. The number of heads for the multi-head attention mechanism for GATH is set as 8.

5.2 Experimental Results

The results of our evaluation experiments are presented in Table 3, 4, and Table 5, Table 6, Table 7, and Table 8, Table 9, Table 10, and Table 11. From these results, we have the following observations:

- (1) In most cases, OC-SVM and Roc-SVM are inferior to other methods. This is because the traditional shallow learning methods do not capture the underlying graph structure information. Besides, we can also find that on the DBLP data set (in Table 6, 7, and 8), when the value of $\%p$ is small, the performance of OC-SVM may be higher than that of GCN_NNPU. This may be because it is difficult for the model to learn good representation features for all nodes when there are rather few positive samples.

Table 3. F1 scores on the Citeseer network using classical positive unlabeled learning methods. Average F1 score and standard deviation are reported for 10 random seeds.

%p	OC-SVM	LINE_OC-SVM	GAE_OC-SVM	Roc-SVM	LINE_Roc-SVM	GAE_Roc-SVM
%0.01	0.023±0.004	0.041±0.011	0.517±0.099	0.018±0.005	0.177±0.036	0.393±0.019
%0.02	0.038±0.011	0.196±0.028	0.614±0.028	0.057±0.007	0.314±0.024	0.464±0.015
%0.03	0.054±0.015	0.304±0.028	0.631±0.024	0.079±0.007	0.381±0.039	0.485±0.020
%0.04	0.090±0.009	0.361±0.024	0.646±0.013	0.115±0.009	0.424±0.026	0.493±0.019
%0.05	0.089±0.017	0.430±0.024	0.658±0.011	0.146±0.016	0.446±0.027	0.500±0.017

Table 4. F1 scores on the Citeseer network using unbiased risk estimator for PU learning. Average F1 score and standard deviation are reported for 10 random seeds. The best results are reported in boldface.

%p	FC_UPU	FS_UPU	GCN_UPU	GAT_UPU	GATH_UPU	LSDAN_UPU
%0.01	0.515±0.112	0.456±0.141	0.337±0.166	0.531±0.083	0.564±0.086	0.647±0.073
%0.02	0.522±0.109	0.553±0.051	0.411±0.139	0.578±0.067	0.606±0.070	0.697±0.067
%0.03	0.533±0.094	0.578±0.026	0.496±0.084	0.582±0.038	0.614±0.042	0.710±0.054
%0.04	0.560±0.060	0.629±0.017	0.540±0.061	0.645±0.017	0.668±0.024	0.717±0.051
%0.05	0.588±0.042	0.681±0.010	0.590±0.055	0.696±0.019	0.711±0.016	0.738±0.036

Table 5. F1 scores on the Citeseer network using non-negative risk estimator for PU learning. Average F1 score and standard deviation are reported for 10 random seeds. The best results reported in boldface.

%p	FC_NNPU	FS_NNPU	GCN_NNPU	GAT_NNPU	GATH_NNPU	LSDAN_NNPU
%0.01	0.684±0.013	0.682±0.007	0.433±0.258	0.775±0.030	0.777±0.029	0.786±0.043
%0.02	0.626±0.054	0.695±0.008	0.564±0.300	0.775±0.024	0.779±0.022	0.804±0.028
%0.03	0.710±0.016	0.705±0.009	0.623±0.259	0.796±0.017	0.803±0.015	0.813±0.014
%0.04	0.734±0.013	0.725±0.009	0.721±0.199	0.814±0.015	0.815±0.009	0.828±0.009
%0.05	0.743±0.015	0.745±0.008	0.812±0.011	0.830±0.011	0.832±0.008	0.840±0.007

Table 6. F1 scores on the DBLP network using classical positive unlabeled learning methods. Average F1 score and standard deviation are reported for 10 random seeds.

%p	OC-SVM	LINE_OC-SVM	GAE_OC-SVM	Roc-SVM	LINE_Roc-SVM	GAE_Roc-SVM
%0.01	0.445±0.029	0.349±0.035	0.576±0.033	0.056±0.012	0.355±0.017	0.515±0.037
%0.02	0.543±0.009	0.471±0.023	0.624±0.017	0.144±0.026	0.463±0.029	0.567±0.038
%0.03	0.580±0.004	0.519±0.018	0.637±0.012	0.234±0.019	0.499±0.022	0.597±0.025
%0.04	0.601±0.006	0.547±0.012	0.643±0.008	0.314±0.037	0.524±0.027	0.613±0.013
%0.05	0.611±0.006	0.567±0.015	0.643±0.007	0.371±0.037	0.545±0.023	0.616±0.017

- (2) Both GAE_OC-SVM and GAE_Roc-SVM outperform OC-SVM and LINE_OC-SVM, Roc-SVM and LINE_Roc-SVM, respectively. This shows the effectiveness of unsupervised graph embedding by using both the adjacency matrix of nodes and the features of node.
- (3) GAT_UPU and GAT_NNPU outperform FC_UPU, FS_UPU, and FC_NNPU, FS_NNPU, respectively. This confirms that it is useful to take node relationships into consideration for node representation learning. Furthermore, GATH_UPU and GATH_NNPU outperform GAT_UPU and GAT_NNPU, which shows the effectiveness of introducing the multi-head attention.

Table 7. F1 scores on the DBLP network using unbiased risk estimator for PU learning. Average F1 score and standard deviation are reported for 10 random seeds. The best results are reported in boldface.

%p	FC_UPU	FS_UPU	GCN_UPU	GAT_UPU	GATH_UPU	LSDAN_UPU
%0.01	0.510±0.108	0.523±0.060	0.398±0.134	0.547±0.033	0.582±0.050	0.687±0.056
%0.02	0.526±0.075	0.612±0.022	0.501±0.096	0.655±0.035	0.690±0.031	0.709±0.067
%0.03	0.572±0.042	0.645±0.012	0.573±0.072	0.729±0.023	0.734±0.018	0.740±0.049
%0.04	0.610±0.029	0.671±0.010	0.633±0.054	0.749±0.021	0.753±0.015	0.757±0.035
%0.05	0.646±0.020	0.693±0.010	0.669±0.052	0.771±0.013	0.771±0.012	0.777±0.033

Table 8. F1 scores on the DBLP network using non-negative risk estimator for PU learning. Average F1 score and standard deviation are reported for 10 random seeds. The best results are reported in boldface.

%p	FC_NNPU	FS_NNPU	GCN_NNPU	GAT_NNPU	GATH_NNPU	LSDAN_NNPU
%0.01	0.650±0.032	0.677±0.007	0.419±0.128	0.767±0.019	0.775±0.018	0.808±0.012
%0.02	0.521±0.092	0.695±0.023	0.599±0.050	0.807±0.017	0.808±0.014	0.833±0.015
%0.03	0.710±0.011	0.715±0.007	0.685±0.032	0.824±0.008	0.825±0.009	0.824±0.008
%0.04	0.597±0.046	0.725±0.011	0.734±0.026	0.836±0.009	0.838±0.009	0.849±0.010
%0.05	0.741±0.009	0.746±0.009	0.760±0.024	0.845±0.009	0.845±0.008	0.857±0.010

Table 9. F1 scores on the Cora network using classical positive unlabeled learning methods. Average F1 score and standard deviation are reported for 10 random seeds.

%p	OC-SVM	LINE_OC-SVM	GAE_OC-SVM	Roc-SVM	LINE_Roc-SVM	GAE_Roc-SVM
%0.01	0.111±0.213	0.101±0.016	0.618±0.046	0.039±0.008	0.229±0.042	0.447±0.040
%0.02	0.263±0.004	0.260±0.031	0.698±0.027	0.073±0.010	0.348±0.032	0.497±0.029
%0.03	0.293±0.010	0.367±0.025	0.700±0.019	0.128±0.013	0.411±0.029	0.504±0.015
%0.04	0.324±0.014	0.433±0.032	0.701±0.018	0.169±0.017	0.443±0.030	0.502±0.017
%0.05	0.358±0.015	0.482±0.026	0.708±0.014	0.218±0.018	0.465±0.017	0.516±0.022

Table 10. F1 scores on the Cora network with the unbiased risk estimator for PU learning. Average F1 score and standard deviation are reported for 10 random seeds. The best results are reported in boldface.

%p	FC_UPU	FS_UPU	GCN_UPU	GAT_UPU	GATH_UPU	LSDAN_UPU
%0.01	0.524±0.098	0.503±0.115	0.454±0.079	0.563±0.080	0.597±0.086	0.746±0.073
%0.02	0.531±0.091	0.567±0.056	0.579±0.059	0.693±0.055	0.720±0.055	0.796±0.050
%0.03	0.561±0.053	0.623±0.024	0.650±0.035	0.771±0.022	0.790±0.017	0.824±0.017
%0.04	0.586±0.036	0.661±0.019	0.686±0.034	0.808±0.020	0.821±0.016	0.836±0.018
%0.05	0.614±0.026	0.686±0.015	0.721±0.021	0.829±0.016	0.838±0.010	0.843±0.013

- (4) The proposed LSDAN_UPU and LSDAN_NNPU outperform GAT_UPU and GAT_NNPU which only capture short-distance neighboring information. The results show the effectiveness of our algorithm in exploiting multi-hop neighbors to capture *long-short distance relationship* in graph learning. Meanwhile, our model shows superior performance in positive unlabeled learning problem.
- (5) The results also show that the proposed LSDAN_UPU and LSDAN_NNPU consistently outperform all the other baselines on all three datasets with different training ratios. It demonstrates that long-short distance aggregation network together with the unbiased and non-negative risk estimators (UPU and NNPU) can better capture data distribution and

Table 11. F1 scores on the Cora network with the non-negative risk estimator for PU learning. Average F1 score and standard deviation are reported for 10 random seeds. The best results are highlighted in boldface.

%p	FC_NNPU	FS_NNPU	GCN_NNPU	GAT_NNPU	GATH_NNPU	LSDAN_NNPU
%0.01	0.542±0.086	0.673±0.008	0.610±0.248	0.772±0.026	0.782±0.013	0.825±0.016
%0.02	0.593±0.039	0.683±0.008	0.771±0.078	0.817±0.024	0.826±0.022	0.841±0.020
%0.03	0.641±0.033	0.697±0.010	0.819±0.041	0.842±0.016	0.848±0.016	0.850±0.013
%0.04	0.666±0.028	0.713±0.010	0.842±0.018	0.859±0.015	0.859±0.010	0.860±0.013
%0.05	0.691±0.025	0.725±0.008	0.850±0.012	0.866±0.008	0.866±0.007	0.867±0.009

the underlying relationship among data by integrating the feature information and graph information into a unified framework.

5.3 Analysis of Different Components

As our proposed model contains two key components: the long-short distance aggregation network (LSDAN) and the positive unlabeled (PU) learning component. In this section, we compare variants of the proposed model with respect to the following aspects to demonstrate the effectiveness of the long-short distance aggregation network and the positive unlabeled (PU) learning component.

The following LSDAN variants are designed for comparison.

- LSDAN $\neg p$: A variant of LSDAN with the positive unlabeled (PU) loss being removed, and only using the cross entropy loss.
- LSDAN_UPU $\neg l$: A variant of LSDAN_UPU with the long-short distance aggregation network being removed, and only using the short-distance aggregation layer.
- LSDAN_NNPU $\neg l$: A variant of LSDAN_NNPU with the long-short distance aggregation network being removed, and only using the short-distance aggregation layer.

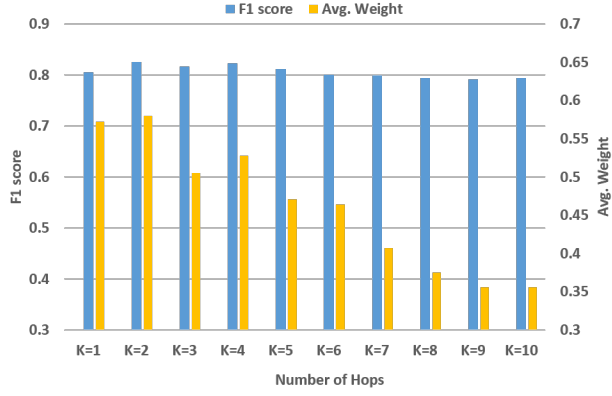
The ablation study results are shown in Table 12, Table 13 and Table 14.

Table 12. F1 score comparisons between LSDAN variants on the Citeseer network.

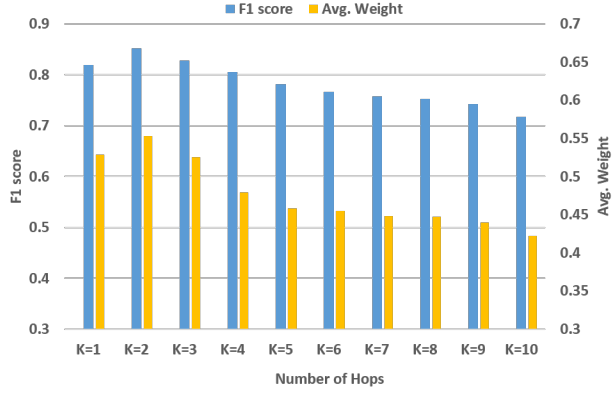
%p	LSDAN $\neg p$	LSDAN_UPU $\neg l$	LSDAN_UPU	LSDAN_NNPU $\neg l$	LSDAN_NNPU
%0.01	0.362	0.531	0.647	0.775	0.786
%0.02	0.373	0.578	0.697	0.775	0.804
%0.03	0.385	0.582	0.710	0.796	0.813
%0.04	0.396	0.645	0.717	0.814	0.828
%0.05	0.409	0.696	0.738	0.830	0.840

Table 13. F1 score comparisons between LSDAN variants on the DBLP network.

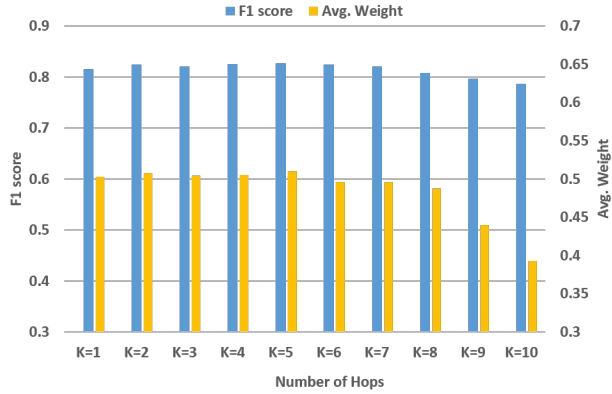
%p	LSDAN $\neg p$	LSDAN_UPU $\neg l$	LSDAN_UPU	LSDAN_NNPU $\neg l$	LSDAN_NNPU
%0.01	0.431	0.547	0.687	0.767	0.808
%0.02	0.437	0.655	0.709	0.807	0.833
%0.03	0.441	0.729	0.740	0.824	0.824
%0.04	0.461	0.749	0.757	0.836	0.849
%0.05	0.472	0.771	0.777	0.845	0.857



(a) Citeseer network



(b) Cora network



(c) DBLP network

Fig. 7. F1 scores and attention weight values by using a single A^k only ($\%p = 0.02$ and $L = 2$).

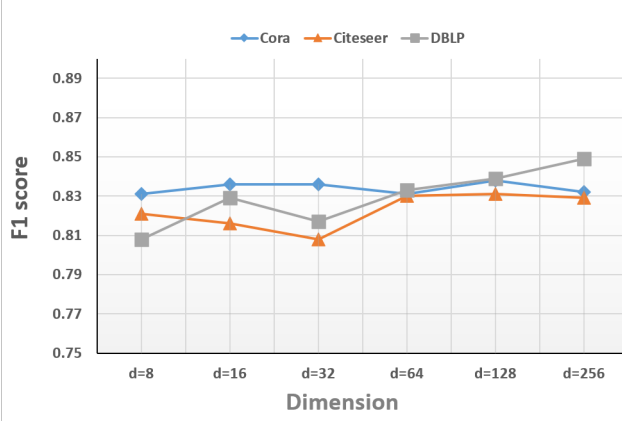
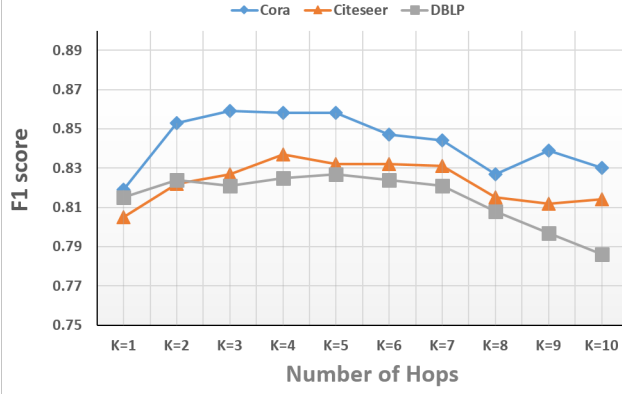
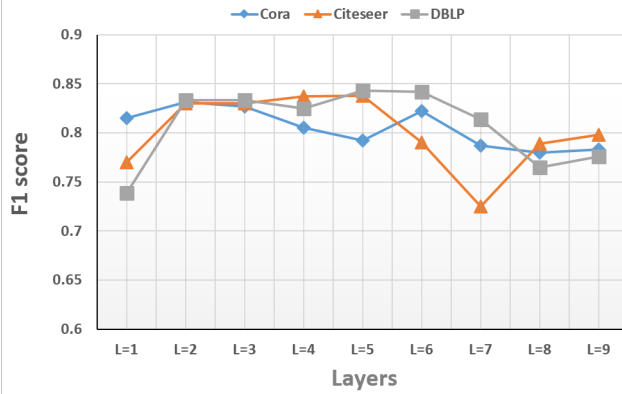
(a) Embedding dimension d (b) Distances at k -hops(c) Layers L Fig. 8. Parameter analysis with respect to embedding dimension d , distance at k -hops, and layers L .

Table 14. F1 score comparisons between LSDAN variants on the Cora network.

%p	LSDAN $\neg p$	LSDAN_UPU $\neg l$	LSDAN_UPU	LSDAN_NNPU $\neg l$	LSDAN_NNPU
%0.01	0.337	0.563	0.746	0.772	0.825
%0.02	0.349	0.693	0.796	0.817	0.841
%0.03	0.372	0.771	0.824	0.842	0.850
%0.04	0.394	0.808	0.836	0.859	0.860
%0.05	0.415	0.829	0.843	0.866	0.867

5.3.1 Impact of the PU loss. We compare LSDAN $\neg p$ with LSDAN_UPU and LSDAN_NNPU to investigate the effectiveness of the positive unlabeled risk estimators employed in our paper. From the result, we find that both LSDAN_UPU and LSDAN_NNPU perform better than LSDAN $\neg p$, which confirms the superiority of the risk estimators for positive unlabeled learning problem.

5.3.2 Impact of long-short distance aggregation network. For fair comparisons, in the actual operation, we only employ one attention head for GAT_UPU and GAT_NNPU. In other words, LSDAN_UPU $\neg l$ are the same as GAT_UPU, and LSDAN_NNPU $\neg l$ are the same as GAT_NNPU. Therefore, the results of LSDAN_UPU $\neg l$ are the the same as that of GAT_UPU, and same for LSDAN_NNPU $\neg l$ vs. GAT_NNPU. In order to verify the effectiveness of the long-short distance aggregation network, we compare LSDAN_UPU with LSDAN_UPU $\neg l$, and LSDAN_NNPU with LSDAN_NNPU $\neg l$. From Table 12, Table 13 and Table 14, we can easily observe that LSDAN_UPU and LSDAN_NNPU perform significantly better than LSDAN_UPU $\neg l$ and LSDAN_NNPU $\neg l$. This confirms that the usage of long-short distance aggregation network can learn a superior representation for nodes in graphs.

5.4 Analysis of the Learned Long-Distance Attention

We further study the learned attentions to understand why our *long-distance attention mechanism* helps improve the performance. Specifically, we examine which graph (A^k) attracts more attentions for the classification task. We report F1 score using A^k only and its attention weight value learned by our LSDAN algorithm. For simplicity, we only report the results on LSDAN_NNPU which are presented in Figure 7.

The results show that the performance of the single graph (A^k) and the long-distance attention value positively correlate. For instance, on the Citeseer dataset, A^2 receives more weight as its predictive power is stronger. The results validate that our approach allows different nodes to focus on the different hops based on the adjacency matrix, to achieve better performance.

5.5 Parameter Analysis

Embedding Dimensions d : We vary d with $\%p = 0.02$, $L = 2$ and report the results on the three datasets in Fig. 8(a). We can find that F1 scores show a clear increase from 8 to 64 on the Cora and DBLP, while it decreases slightly in the 32nd dimension in the Citeseer. When the number of embedding dimensions continuously increases, the performance starts to remain stable. This is intuitive as more embedding dimensions can encode more useful information from data.

Distance at κ -Hops : We also report F1 scores over different choices of κ with $\%p = 0.02$ and $L = 2$ on the three datasets in Fig. 8(b). It can be seen from Fig. 8(b) that when $k=4$, satisfactory results have been achieved on different data sets. This confirms that the long distance relation is really important to better capture graph structure information, and multiple graphs can learn complementary local information. Simultaneously, when k is greater than 4, as k gets bigger and bigger, the result becomes worse, possible because some redundant information interfere

the learning, and the learned k -hop relational information becomes less informative for node representation learning.

Number of layers L : Fig. 8(c) shows the influence of the number of layers on performance on three datasets. Here, we set $\kappa = 4$, and $\%p = 0.02$. For the datasets considered here, best results are obtained with a 2- or 3-layer model. We can see that the setting $L = 2$ has a significant improvement over the setting $L = 1$ on three datasets. We observe that the performance will slightly decrease for models deeper than 6 layers, this may overfit as the number of parameters increases with model depth.

6 DISCUSSION

PU learning is traditionally applied to data with independent and identical distributions (*i.e.* non-relational data). Some earlier works have extended it to graph (relational data) databases. For example, [51] proposed an integrated approach to select discriminative features for graph classification based upon positive and unlabeled graphs. [37] proposed a learning framework for classifying a bag of multiple graphs. Therefore, there are existing work which apply PU learning to graphs/networks, but under different learning settings.

In our problem settings, although nodes are not independent, we can assume that labels are independently provided for selected nodes (which will be marked as labeled nodes). In other words, when setting a set of nodes to be labeled, a random approach is used to select a small portion of nodes being labeled. In fact, all existing works in node classification (transductive graph learning) employ this setting. Once the labels are provided, graph neural networks here are employed to learn the new representation for each node. After each node is represented into a vector space, this task is similar to general PU learning, and then the risk estimators can be employed for PU learning.

One novelty of our approach is that we integrate the feature learning by GNNs and risk estimation into a unified and end-to-end framework. As demonstrated in our algorithm, this approach performs very well in the benchmark datasets.

In this paper, we are following common protocols in the research to set up the PU learning for graphs, with a small portion of randomly selected nodes being labeled. However, some nodes may have a higher chance being labeled, due to their linkages or connections. This is, indeed, determined by the applications and network structures. It will be a very interesting direction for future work to investigate further.

7 CONCLUSION

In this paper, we propose a novel long-short distance aggregation network (LSDAN) for positive unlabeled graph learning. We argue that existing algorithms largely overlook the *long-distance relationship*, and only exploit 1-hop neighbors to aggregate information to learn feature representation for nodes. In order to leverage long-distance relation between nodes, we propose a long-short distance aggregation network to jointly exploit the short-distance and long-short attention from different range of neighborhood to learn feature for each node. In addition, two novel risk estimators are proposed for positive unlabeled graph learning. Experiments and comparisons on three benchmark graph datasets demonstrate the effectiveness of our algorithm.

ACKNOWLEDGMENTS

This research is supported by the U.S. National Science Foundation (NSF) through Grant Nos. IIS-1763452, CNS-1828181, and IIS-2027339.

REFERENCES

- [1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proc. of ICLR*.
- [2] Shaosheng Cao, Wei Lu, and Qionghai Xu. 2016. Deep neural networks for learning graph representations. In *Proc. of AAAI*. 1145–1152.
- [3] Meihao Chen, Zhuoru Lin, and Kyunghyun Cho. 2017. Graph Convolutional Networks for Classification with a Structured Label Space. In *arXiv preprint arXiv:1804.08049*.
- [4] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. *Advances in Neural Information Processing Systems* 33 (2020).
- [5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. of NIPS*. 3844–3852.
- [6] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Proc. of NIPS*. 2224–2232.
- [7] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Proc. of NIPS*. 6530–6539.
- [8] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. *arXiv preprint arXiv:1905.05178* (2019).
- [9] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proc. of SIGKDD*. ACM, 855–864.
- [10] Will Hamilton, Zhitaoying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proc. of NIPS*. 1024–1034.
- [11] William L. Hamilton, Zhitaoying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. of NIPS*. 1024–1034.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*. 770–778.
- [13] Mikael Henaff, Joan Bruna, and Yann Lecun. 2015. Deep Convolutional Networks on Graph-Structured Data. In *arXiv preprint arXiv:1506.05163*.
- [14] Shantanu Jain, Martha White, and Predrag Radivojac. 2016. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Proc. of NIPS*. 2693–2701.
- [15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [16] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).
- [17] Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. 2017. Positive-unlabeled learning with non-negative risk estimator. In *Proc. of NIPS*. 1675–1685.
- [18] Ryuichi Kiryo, Gang Niu, Marthinus C. Du Plessis, and Masashi Sugiyama. 2017. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In *Proc. of NIPS*. 1674–1684.
- [19] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive Graph Convolutional Neural Networks. In *Proc. of AAAI*. 3546–3553.
- [20] Xiaoli Li and Bing Liu. 2003. Learning to classify texts using positive and unlabeled data. In *Proc. of IJCAI*. 587–592.
- [21] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *Proc. of ICLR*.
- [22] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. 2003. Building Text Classifiers Using Positive and Unlabeled Examples. In *Proc. of ICDM*. 179.
- [23] Gang Niu and Masashi Sugiyama. 2015. Convex formulation for learning from positive and unlabeled data. In *Proc. of ICML*. 1386–1394.
- [24] Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. 2020. Learning graph embedding with adversarial training methods. *IEEE Transactions on Cybernetics* 50, 6 (2020), 2475–2487.
- [25] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407* (2018).
- [26] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. In *Proc. of IJCAI*. 1895–1901.

- [27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proc. of SIGKDD*. ACM, 701–710.
- [28] M. C. Du Plessis, Gang Niu, and Masashi Sugiyama. 2014. Analysis of Learning from Positive and Unlabeled Data. In *Proc. of NIPS*. 703–711.
- [29] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Semi-supervised User Geolocation via Graph Convolutional Networks. In *Proc. of ACL*. 2009–2019.
- [30] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80.
- [31] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2014. Estimating the Support of a High-Dimensional Distribution. *Neural Computation* 13, 7 (2014), 1443–1471.
- [32] Xiao Shen and Fu-Lai Chung. 2017. Deep network embedding with aggregated proximity preserving. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 40–43.
- [33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proc. of WWW*. 1067–1077.
- [34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [35] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *Proc. of IJCAI*. 3670–3676.
- [36] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proc. of SIGKDD*. ACM, 1225–1234.
- [37] Jia Wu, Shirui Pan, Xingquan Zhu, Chengqi Zhang, and Xindong Wu. 2017. Positive and Unlabeled Multi-Graph Learning. *IEEE Trans. Cybernetics* 47, 4 (2017), 818–829.
- [38] Man Wu, Shirui Pan, Lan Du, Ivo W. Tsang, Xingquan Zhu, and Bo Du. 2019. Long-short Distance Aggregation Networks for Positive Unlabeled Graph Learning. In *Proc. of ACM CIKM International Conference*.
- [39] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. 2020. Unsupervised Domain Adaptive Graph Convolutional Networks. In *Proceedings of The Web Conference 2020*. 1457–1467.
- [40] Man Wu, Shirui Pan, Xingquan Zhu, Chuan Zhou, and Lei Pan. 2019. Domain-adversarial graph neural networks for text classification. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 648–657.
- [41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
- [42] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).
- [43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
- [44] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. 2018. On exploring semantic meanings of links for embedding social networks. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 479–488.
- [45] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information.. In *Proc. of IJCAI*. 2111–2117.
- [46] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. 2018. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *Proc. of NIPS*. 6412–6422.
- [47] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design space for graph neural networks. *Advances in Neural Information Processing Systems* 33 (2020).
- [48] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*. 11983–11993.
- [49] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *Proc. of UAI*. 339–349.
- [50] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks.. In *Proc. of IJCAI*, Vol. 18. 3155–3161.
- [51] Yuchen Zhao, Xiangnan Kong, and Philip S Yu. 2012. Positive and Unlabeled Learning for Graph Classification. In *Proc. of ICDM*. 962–971.