GLDL: Graph Label Distribution Learning

Yufei Jin[1], Richard Gao[2], Yi He[3], Xingquan Zhu[1]

[1]Dept. of Electrical Engineering & Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA
[2]Dept. of Computer Science, Rice University, Houston, TX 77005, USA
[3]Dept. of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
yjin2021@fau.edu; rdg3@rice.edu; yihe@cs.odu.edu; xzhu3@fau.edu

Abstract

Label Distribution Learning (LDL), as a more general learning setting than generic single-label and multi-label learning, has been commonly used in computer vision and many other applications. To date, existing LDL approaches are designed and applied to data without considering the interdependence between instances. In this paper, we propose a Graph Label Distribution Learning (GLDL) framework, which explicitly models three types of relationships: instance-instance, label-label, and instance-label, to learn the label distribution for networked data. A label-label network is learned to capture label-to-label correlation, through which GLDL can accurately learn label distributions for nodes. Dual graph convolution network (GCN) Co-training with heterogeneous message passing ensures two GCNs, one focusing on instance-instance relationship and the other one targeting label-label correlation, are jointly trained such that instance-instance relationship can help induce label-label correlation and vice versa. Our theoretical study derives the error bound of GLDL. For verification, four benchmark datasets with label distributions for nodes are created using common graph benchmarks. The experiments show that considering dependency helps learn better label distributions for networked data, compared to state-of-the-art LDL baseline. In addition, GLDL not only outperforms simple GCN and graph attention networks (GAT) using distribution loss but is also superior to its variant considering label-label relationship as a static network. GLDL and its benchmarks are the first research endeavors to address LDL for graphs. Code and benchmark data are released for public access.

Introduction

Label distribution learning (LDL) enables the assignment of a distribution to the label of each instance, quantitatively representing the *description degree* of the label to the instance (Geng, Yin, and Zhou 2013; Chen et al. 2020). As such, LDL advances the traditional single/multi-label learning's aim from answering the question of "can this/these label(s) describe the instance?" (*i.e.* binary answers) to "how well a label characterizes the instance?" (*i.e.* numeric answers) (Carbonell, Michalski, and Mitchell 1983; Kotsiantis, Zaharakis, and Pintelas 2006; Zhang et al. 2021; Chen et al. 2019; Xie et al. 2023).

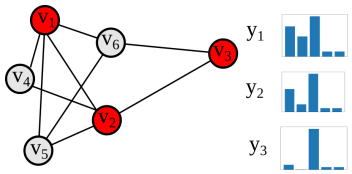

Figure 1: A motivating example of graph label distribution learning, where the label of each colored (labeled) node is a distribution, influenced by its position within the network. Class 3 is the dominant label for nodes v_1, v_2, and v_3. For v_3, Class 3 demonstrates a weaker correlation to other classes (*e.g.*, a company with a focused business); in contrast, for v_1 and v_2, Class 3 exhibits a stronger correlation to other classes (*e.g.*, a company with more diverse business scope).

Enabling the modeling of label distribution allows LDL to have a richer and more nuanced description of the underlying objects, invaluable for many computer vision (CV) and natural language processing (NLP) applications. For example, in an image portraying a lake surrounded by trees with a mountain in the distance, LDL aptly captures the essence of the scene by indicating that the lake is the most prominent element in terms of pixel coverage, while the mountain, being less visible, holds a minor presence (Xu et al. 2023).

Despite its success across various domains, LDL has primarily been applied to data characterized by independent and identically distributed (*i.e.* IID) properties. Yet, recent advances in social networks and complex information systems have resulted in a large number of applications where data convey dependency relationships. Consider the task of analyzing a business collaboration network, as shown in Fig. 1, where nodes represent companies and edges signify shared users or customers. A company may offer a variety of services with different levels of emphasis (*e.g.*, a bike store might also provide rental or coaching services). Predicting the companies' business scopes in this network can be conceptually cast as an LDL task. The Yelp and Yelp2 datasets explored in our experiments present similar applications, where the label distribution of a node (*i.e.* a restaurant) reflects the diverse ratings from customers. In this case, label distribution modeling is more informative than a single aggregate rating, as it provides a richer understanding

of customer opinions and preferences. However, applying standard LDL methods to such network data would require careful consideration, as they tend to miss out the relational information embedded within the topological structure.

Whereas more recent LDL methods have considered label-label correlation in the modeling process (Jia et al. 2018; Wang and Geng 2023; Kou et al. 2023), they still postulate IID instances. For graphs, a label-label relationship is also impacted by network topology, *i.e.* labels of connected nodes may have a stronger correlation, as shown in Fig. 1. Applying LDL to graphs presents distinct challenges due to the non-IID nature of graph data, described as follows.

- First, network topology plays a significant role in determining label distributions. The topological features convey learnable semantic meanings, such as density, degree, reachability, etc. The label distribution for a node is not only based on its contents but also heavily influenced by its position within the network.
- Second, label distribution of a node can be affected by its neighboring nodes, which may lead to inconsistency and disagreement between node features and topological structure. Balancing and integrating these potentially conflicting sources of information is challenging yet was not encountered in traditional LDL settings.
- Third, unlike traditional LDL that mostly models label-label correlation across the whole dataset, in graphs, a node's label distribution can be influenced by its neighbors, leading to varied label-label dependencies across different local regions of the graph. This requires a new design that can capture and harmonize both local and global label-wise dependencies.

Due to these challenges, simply extending the existing graph learning models, such as graph convolutional networks (GCNs) or graph attention networks (GATs), to LDL by using a distributional loss (*e.g.*, KL-divergence) is suboptimal because label correlation is overlooked.

Motivated by this, we propose a new learning framework, termed *graph label distribution learning* (GLDL). Our key idea is to induce a label-label network and combine it with node-node network in the graph learning process. The learning of node-node network aims to obtain good node feature representations, and the learned node features will help induce a label-label network, whose results will in turn improve the learning of node embeddings. The two networks are collaboratively learned, ensuring the resultant node features can jointly minimize the distributional loss in the label space and the topology loss of the induced label-label network. The technical merits of our GLDL are backed up by theoretical analysis and empirical studies. Furthermore, our experiments show that GLDL has significantly better stability and robustness in tackling over-smoothness, a common phenomenon observed for GCN learning, especially for networks with severely imbalanced label distributions (*i.e.* one or two labels dominating the whole dataset).

Specific contributions of this paper are as follows:

1. This is the first study to explore the label distribution learning (LDL) problem for graphs. The technical challenges of this problem encompass a complex interplay among graph topology, node features, and label correlations, with details unfolded in Section .
2. We propose a new GLDL approach that employs static and dynamic strategies for effective label correlation modeling. At its core, GLDL jointly learns node and label embeddings, aiming for a globally optimized representation. Technical details are presented in Section .
3. We provide an in-depth theoretical analysis, deriving the generalization error bounds of our proposed GLDL in Section 29. This lays a rigorous theoretical bedrock for ensuing exploration in the domain of graph LDL.
4. Extensive experiments are carried out to substantiate the viability and effectiveness of our proposed approach. Results and findings are documented in Section 29.
5. Our code, benchmark data, and supplementary material are openly accessible at GitHub[1].

Related Work

Label Distribution Learning (LDL) Existing LDL methods mainly fall into three categories: problem transformation (PT), algorithm adaption (AA), and specialized algorithm (SA). In the realm of PT, (Geng 2016) and (Borchani et al. 2015) reconceptualized the LDL challenge as a single-label learning task, where label probabilities are harnessed as weights. On the other hand, AA methods repurpose established classifiers to fit the LDL milieu. Notably, AA-kNN (Geng 2016) leverages the instance-neighbor distances as heuristics to approximate label distributions. Turning our attention to SA methods, most existing LDL methods employ SA design. LDLLC (Jia et al. 2018) encodes the label correlation into a distance to measure the similarity of any two labels. A Gaussian label distribution learning method (Xu et al. 2023) employs a parametric model underpinned by an optimization strategy assessing KL-divergence distance between Gaussian distributions. Note, these prevalent LDL methods assume IID data, and often neglect to capture instance-wise correlations. Existing studies (Geng 2016) vouch for the superior performance of SA over its LDL counterparts. Consequently, in our exploration, we juxtapose SA against other LDL methods to discern its adaptability and efficacy, especially in the intricate landscape of networked data (graphs).

Graph Learning (GL) Graph neural networks (GNNs) have solidified their stature as a cornerstone model for graph learning (GL) and data mining on graphs. Central to GNNs are two fundamental stages: neural message passing and aggregation. The aggregation stage synthesizes information from neighboring nodes to refine embeddings for the current node. Multiple GNN variants have emerged, each offering nuanced interpretations and expansions. To wit, Graph Convolution Network (GCN) (Kipf and Welling 2017) inspired by the graph spectral theory capitalizes on the eigen-decomposition of the graph Laplacian matrix. Graph At-

[1]https://github.com/Listener-Watcher/Graph-Distribution-Learning.

tention Network (GAT) (Veličković et al. 2018) is adept at learning weights attributed to neighboring nodes during aggregation. Graph Isomorphism Network (GIN) (Xu et al. 2019) introduces a neural network as its aggregation function, taking cues from the Weisfeiler-Lehman test, which aims to accentuate discrepancies between disparate graphs. GraphSAGE (Hamilton, Ying, and Leskovec 2017) adopts a sub-sampling approach for neighbor nodes, ensuring a more agile and scalable training regimen tailored for extensive graphs. Despite node classification emerging as a prevalent application for these GL models, a significant oversight is their treatment of labels as isolated entities. This tunnel vision fails to recognize potential correlations between labels, resulting in suboptimal generalization in LDL scenarios. In this exploration of GLDL, a pioneering foray into addressing the LDL challenge within graph contexts, we have elected to use GCN as our backbone model, as its streamlined architecture not only facilitates implementation but also serves as an effective scaffold for our innovative contributions.

Preliminaries

Notation Appointment We follow graph learning conventions. Let $G = (V, E, X, Y)$ denote a graph, where $V = \{v_i\}_{i=1,\cdots,n}$ is the vertex set representing nodes of the graph G, and E is the edge set. Denoted by $e_{ij} = (v_i, v_j) \in E$ is the edge linking node v_i and node v_j. The graph topology (V, E) is encoded in an adjacency matrix $A \in \{0, 1\}^{n\times n}$, where $A_{i,j} = 1$ if $e_{ij} \in E$ and $A_{i,j} = 0$ otherwise. Let $\Delta_i = \{v_j | e_{ij} \in E, \forall j\}$ denote the neighbors of node v_i, where E_i includes the set of edges incident to node v_i, *i.e,* $E_i = \{e_{ij} | v_j \in \Delta_i, \forall j\}$. To ease derivation, we write $\bar{A} = A + I$ the adjacency matrix with a self-loop added on each node, and $\bar{D}$ is the diagonal matrix of $\bar{A}$.

Problem Statement Let $X \in \mathbb{R}^{n\times m}$ denote the feature matrix associating with n nodes, where each node v_i is described by an m-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^m$. In our problem of graph label distribution learning, the goal is find a mapping $\psi : (G, X) \mapsto Y$, where $Y \in \mathbb{R}^q$ is a distribution of descriptive labels over q classes. Namely, $y_{i,j} \in [0, 1]$ denotes the probability that the node v_i belongs to the j-th class, and $\sum_{j=1}^{q} y_{i,j} = 1$. In this study, we frame the learning problem of ψ in a transductive regime (Bacciu et al. 2020), where the ground-truth label distributions are available for a node subset $V_{tr} \subset V$ during training. $|V_{tr}| << |V|$. The learned ψ is expected to generalize well at the remaining unlabeled node subset $V \setminus V_{tr}$.

Graph Convolution Network GCN is a transductive GL model proposed by (Kipf and Welling 2017). The crux of GCN is to propagate node information through neighbors. One GCN layer at depth i can be formulated as:

$$H^i = \bar{D}^{-\frac{1}{2}} \bar{A} \bar{D}^{-\frac{1}{2}} H^{i-1} W^i, \tag{1}$$

where H^i is the node representations at layer i, with the initial embedding features being each node's attributes, i.e. $H^0 = X$, and W^i is the learnable parameter for layer i. Multiple GCN layers can be stacked and used to integrate topology and features into a hidden dimension embedding.

Technical Challenge and Our Thoughts Traditional label distribution learning focuses on inducing ψ from IID data. However, when addressing networked data, one is confronted with an intricate task: balancing two disparate graph signals within a unified learning objective. The first signal originates from the nodal contents X, whereas the second is intrinsically encoded within the topological structure A. The complexity of this task is accentuated within an LDL framework. Unlike conventional learning paradigms, where targets often stand as independent variables, in LDL they manifest as closely intertwined descriptive labels. This introduces multifaceted dynamics. For instance, if node i exhibits a high probability of belonging to class j, it conversely signals a diminished likelihood of its association with other classes. Yet, complicating this further is the influence exerted by its immediate neighbors Δ_i on the graph. This relationship stems from the principle of graph homophily (Zhu et al. 2020), which posits that directly connected nodes exhibit a propensity to converge in their label characteristics. As such, a competent graph LDL learner requires a dedicated optimization objective that can concurrently respect and capture three information channels: the inherent characteristics of the nodes, the intricate network of graph topology, and the nuanced interplay of label correlations.

To tailor such objective, our key idea is to build a label-label network $G^c = (V^c, E^c, X^c, Y^c)$, in which each node $v_i^c \in V^c$ represents a class label, and $|V_c| = q$. Each edge $e_{i,j}^c \in E^c$ captures the correlation between the two connecting classes i and j. The topology of G^c thus models the dependency structure in the label space. The semantic meaning of each class i is captured by its node embedding $x_i^c \in X^c$. Each label node i is assigned a hard label y_i^c representing the class i in one-hot code. The main objective is to learn two GCNs over G and G^c jointly, and then induce the bipartite edges connecting nodes in V and V^c, where each such edge $\hat{e}_{i,j}$ indicates the probability that node v_i belongs to the class node v_j^c. We present technical details in the next section.

GLDL: The Proposed Method

Overall Framework

The overall framework of our proposed Graph Label Distribution Learning (`GLDL`) approach is illustrated in Figure 2, which mainly contains the learning processes of two interlinked networks. The input graph G is the node-node network (lower panel), on which we train a GCN for node representation and predicting the nodal label distributions. The label-label network (upper panel) is constructed from the nodes in G that possess ground-truth distributions. This is realized via a Graph Generation function $\mathrm{GG}(\cdot)$, which also enables message passing between the two networks. The overarching learning objective of `GLDL` comprises three key components as follows. 1) The design and training of an LDL learner that strives to minimize the KL-divergence loss amongst labeled nodes within the node-node network. 2) The formulation and training of a multi-class learner for the label-label network, aiming to distill the intrinsic semantic relationships and interdependencies present within the label space, with the objective being the reduction of cross

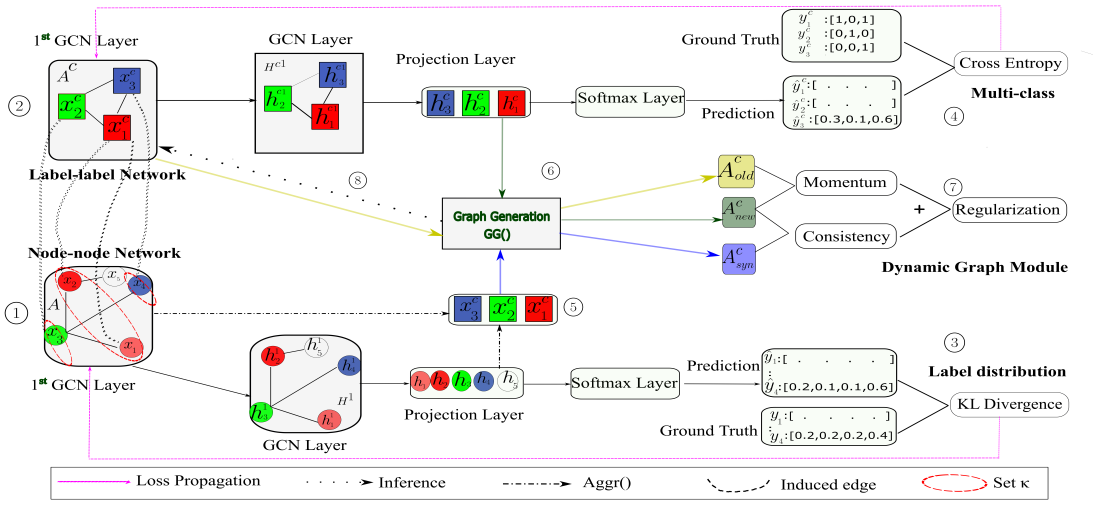

Figure 2: The proposed GLDL framework. Lower panel denotes node-node network, and upper panel denotes label-label network, which is induced through the Graph Generation GG() process as in Eq 4. For both networks, colored nodes are labeled. Brief process: A node-node network in ① is first used to induce label-label network in ②. The dual GCN training at ③ and ④ will result in respective node embedding. Steps ⑤ and ⑥ will each induce a network regulated by consistency and momentum terms in ⑦. Step ⑧ updates the label-label network for succeeding training process ($q = 3$ in this example).

entropy. 3) The regularization terms ensure the continuity of the construction of the label-label network, devoid of sudden topological shifts.

Label-Dependency-Aware Graph Generation

To construct the label-label network G^c, the feature matrix of the label nodes $X^c \in \mathbb{R}^{q\times m}$ is initialized based on the labeled nodes in G that belongs to each corresponding class. Specifically, for class j, its feature vector x_j^c aggregates the labeled nodes of which the dominant class is j, defined as

$$x_j^c = \text{Agg}(\kappa) = \frac{1}{|\kappa|}\sum_{i\in\kappa} x_i, \tag{2}$$

where κ:={ $v_i \mid \arg\max(Y_i) = j, v_i \in V$ }. To build the edges E^c, a straightforward idea is to leverage the similarity metrics such as Pearson correlation, leading to the ***static*** label-label network, with the generation function defined as:

$$A_{i,j}^{c\text{-static}} := GG(x_i^c, x_j^c) = \frac{1}{1+\exp^{-\langle x_i^c, x_j^c\rangle}}, \tag{3}$$

where the (i, j)-th entry of A^c gauges the inner-product similarity between the two class nodes v_i^c and v_j^c. We note that the design of Eq. (3) takes an analogical form as the decoder layer of a non-probabilistic graph autoencoder (Kipf and Welling 2016). As the label-label network is constructed only once before learning starts, the heuristic of Eq. (3) can tame the training of GCN upon it and avoid loss oscillation incurred by the topological changes of G^c.

While this static generation strategy offers simplicity, it introduces potential pitfalls. On the one hand, such a static approach might inadvertently infuse noise, particularly if the initialization of label nodes is ill-conditioned. Such perturbations can impede the accurate representation and correlation of labels. On the other hand, the static strategy appears to be myopic in its design, neglecting the dynamic interplay between the embeddings of the original node-node network and the label distribution. Given the intricate nature of graph data, even slight variations in node embeddings can lead to precipitate substantially different label-label networks. This necessitates a more dynamic and adaptive strategy that capitalizes on the evolving information during the learning process, thereby constructing a more informed and adaptive label-label network. By doing so, it brings Expectation-Maximization style improvement to benefit both node representation and label correlation.

For ***dynamic*** label-label network construction, we draw insights from graph condensation (Jin et al. 2022) to learn a parametric graph generator $GG(\cdot)$, defined as:

$$\begin{aligned} A_{i,j}^{c\text{-dynamic}} &:= GG(x_i^c, x_j^c) \\ &= \text{sigmoid}\Big(\frac{MLP_\Phi([\mathbf{x}_i^c;\mathbf{x}_j^c]) + MLP_\Phi([\mathbf{x}_j^c;\mathbf{x}_i^c])}{2}\Big), \end{aligned} \tag{4}$$

with Φ denoting learnable parameters of multi-layer perceptron (MLP), and $[\cdot;\cdot]$ indicates concatenation. In this dynamic strategy, a new graph G^c is re-generated regularly (e.g., based on a fixed number of epochs).

Dual GCN Co-Training via Heterogeneous Message Passing

After having the node-node network G and the label-label network G^c, the next question would be to train the GCN model that allows message passing between the two graphs,

so that the learned node representations can capture the dependency structures in both topological and label spaces.

To that end, we first define the objective function of training GCN model on each graph independently, and then devise the mechanism to bolster message passing between the two graphs having heterogeneous nodes.

To train GCN on G, the Kullback-Leibler (KL) divergence presents itself as an apt loss function. This metric gauges the divergence between the predicted and ground-truth nodal label distributions. Formally, the KL-divergence for a node v_i is defined as:

$$\ell_{KL}(v_i) = \sum_{j \in [1,q]} y_{i,j} \times \log \frac{y_{i,j}}{\hat{y}_{i,j}}, \tag{5}$$

where $\hat{y}_{i,j}$ denotes the predicted likelihood that node i belongs to class j. Figure 2 demonstrate an example in which the ground truth $y_{4,3} = 0.2$ and the predicted $\hat{y}_{4,3} = 0.1$.

To train GCN^c on the label-label network G^c, a cross-entropy loss is employed. This ensures the mapped representation of a learned label node accurately corresponds to its class. For a single label node i, the loss is defined as:

$$\ell_{CE}(v_i^c) = \sum_{j \in [1,q]} y_{i,j}^c \times \log \hat{y}_{i,j}^c \tag{6}$$

where $y_{i,j}^c$ is the ground truth label, represented as a scalar of index j of the one-hot vector y_i of length q, and $\hat{y}_{i,j}^c$ signifies the estimated probability for label node i at index j. An illustrative example in Figure 2 showcases $y_{3,3}^c = 1.0$ while $\hat{y}_{3,3}^c = 0.6$.

The linchpin of our GLDL approach is the ability to facilitate message passing between the divergent graphs G and G^c. This is achieved by forming induced edges based on the training node labels Y. Specifically, an induced edge materializes between node v_i and a label node v_j^c if $\arg\max(y_i) = j$. This ensures proper communication between label-label correlation and node embedding learning. During the initial convolution layer, edges spanning across both graphs G and G^c are meticulously considered, allowing information exchange between the two learning processes. After a fixed number of epochs, both node features and label features undergo an update, adopting the learned embeddings H and H^c, as visualized in Figure 2. The epoch frequency designated for updating node features is denoted as $freq_v$, while that for label features is $freq_c$. For effectiveness without losing simplicity, we synchronize the label feature update with the graph update, meaning we contemporaneously update the label graph G^c whenever the label feature X^c undergoes an update.

Expedited Graph Training with Momentum and Consistency Regularization

A key difficulty in training the dual GCN model is ensuring the stability of the training process. Rapid fluctuations or oscillations in the learning process can make convergence difficult and slow. To tame the dynamically generated G^c, two unsupervised regularization terms are proposed to ease the GCN training difficulty. The parametric graph generator $GG(X^c)$ is updated periodically after several epochs, using X^c as initial label embeddings and demonstrated as H^c in Figure 2. This aids in the training of the network and consequently drives the graph induction. The resultant synthetic graph A_{syn}^c is derived from an aggregation of the node embedding features, while A_{new}^c arises from the learned label embedding, calculated by $A_{syn}^c = GG(\text{Agg}(H))$ and $A_{new}^c = GG(H^c)$. The two regularization terms are formally defined as follows.

Consistency Loss intermediately quantifies the discrepancy between the learned node embedding and the label embedding through the constructed graph topology, defined as:

$$\text{consistency} = ||\Lambda(D_{new}^c - A_{new}^c) - \Lambda(D_{syn}^c - A_{syn}^c)||_2^2, \tag{7}$$

where instead of the adjacency matrices, we resort to the eigenvalues of the graph Laplacian $\Lambda(D_{new}^c - A_{new}^c)$ to delineate the variances between graphs. This strategy ensures that the properties of the two graphs remain analogous, insulating against complications such as graph isomorphism. The intuition behind this loss is that, when the predictions are consistent with the given label distributions, the learning process is more aligned, and can lead to faster convergence. This loss helps the model stay on track by penalizing significant deviations from expected representations.

Momentum Loss ensures that there are no drastic changes between consecutive epochs by modulating the rate of evolution of the graph, defined as:

$$\text{momentum} = ||\Lambda(D_{new}^c - A_{new}^c) - \Lambda(D_{old}^c - A_{old}^c)||_2^2, \tag{8}$$

where A_{new}^c and A_{old}^c represent the graph generated from current epoch embedding H^c and that from the previous epoch, respectively. By modulating the fluctuations in consecutive graph formations, it fosters stability during model training. The holistic regularization objective combines these two losses as regularization $=$ consistency $+$ $\alpha \times$ momentum, offering a tunable balance α between them.

Algorithm

Algorithm 1 reports detailed steps and static and dynamic network details.

Label-Label Graph Module The GCN^c architecture for label graph G^c consists of GCN layers, projection layers, and a softmax layer. The 1st GCN layer integrates node information from both label-label graph G^c and node-node graph G following Eq. (1) and line 11 in Algorithm 1. The rest of the GCN layers integrate node information from nodes within graph G^c alone. The learned hidden label embedding $H^c = [h_1^c; \cdots; h_q^c]$ is then forward projected to the original feature space to ensure that the label representation can be aligned with the node features. This alignment process is crucial to allow node propagation between graphs in the 1st GCN layer. The learned label representation is also used to regularize the dynamic graph generator $GG(H^c)$ as part of the momentum loss for A_{new}^c in Fig 2 and line 25 in Algorithm 1. A softmax layer is applied to obtain probability values needed for computing the cross-entropy loss in Eq. (6) and line 14 in Algorithm 1. The learned embedding is updated as the new label features for a certain frequency denoted by freq_c following line 22 in Algorithm 1.

Node-Node Graph Module The GCN architecture for node graph G is the same as the GCN^c except for the number of nodes, independent learnable weight, and final objective function. After applying the softmax function, KL-divergence loss as stated in Eq. (5) is used as the loss function. The learned embedding is updated using a different frequency denoted by freq_v following line 19 in Algorithm 1. The updated embedding $H = [h_1; \cdots; h_n]$ is also used to regularize the dynamic graph generator $GG(Agg(H))$ as part of the consistency loss for A^c_{syn} following line 25.

Dynamic Graph Module This module is an optional module to replace the static graph generation function $GG(\cdot)$ as stated in Eq. (3). Following Eqs. (7), (8) or from lines 23 to 27 in Algorithm 1, the parametric model is trained every certain epoch (same frequency freq_c as the label features are updated for simplicity of the model) and then infer a new graph topology A^c_{new} to replace the original A^c_{old} for succeeding training.

Algorithm 1: The GLDL algorithm

Input: G:(A,X,Y), freq_v, freq_c, epochs, epochs_{GG}
Model: GCN,GCN^c, GG($\cdot$)
Init: $A^h \in \mathbb{R}^{(m+q,m+q)} \leftarrow 0$, $E^h \in \mathbb{R}^{(m,q)} \leftarrow 0$
Output: $\hat{Y}$

1 $X^c \leftarrow Agg(X_{train})$; $A^c \leftarrow GG(X^c)$
2 **for** $i \leq m$ **do**
3 **for** $j \leq q$ **do**
4 **if** $\text{argmax}(Y_i) = j$ **then**
5 $E^h_{i,j} \leftarrow 1$
6 $A^h \leftarrow \begin{bmatrix} A & E^h \\ (E^h)^T & A^c \end{bmatrix}$
7 $X^h \leftarrow [X\ ;\ X^c]$; $X^{hc} \leftarrow [X\ ;\ X^c]$
8 **for** $i \leq$ *epochs* **do**
9 $H^1 \leftarrow \text{GCN}^{(1)}(A^h, X^h)$
10 $\hat{Y} \leftarrow \text{GCN}(A, H^1)$
11 $H^{c(1)} \leftarrow \text{GCN}^{c(1)}(A^h, X^{hc})$.embedding()
12 $\hat{Y}^c \leftarrow \text{GCN}^c(A^c, H^{c1})$
13 $\ell_{KL}(Y, \hat{Y}) \leftarrow$ compute loss using Eq. (5)
14 $\ell_{CE}(Y^c, \hat{Y}^c) \leftarrow$ compute loss using Eq. (6)
15 compute gradient of $\ell_{KL}$and ℓ_{CE}
16 update GCN and GCN^c with gradient
17 **if** $i \,\%\, freq_v == 0$ **then**
18 $H \leftarrow$ GCN.embedding()
19 $X^h \leftarrow [H\ ;\ X^c]$
20 **if** $i \,\%\, freq_c == 0$ **then**
21 $H^c \leftarrow \text{GCN}^c$.embedding()
22 $X^{ch} \leftarrow [X\ ;\ H^c]$
23 **if** *dynamic mode* **then**
24 **for** $j = 1, 2, \ldots,$ *epochs*$_{GG}$ **do**
25 Train $GG(\cdot)$ with Eqs. , (7) , (8)
26 $A^c \leftarrow GG$.inference(H^c)
27 update A^h with A^c
28 $\hat{Y}^c \leftarrow GCN^c(A^h, A^c, X^h, X^c)$
29 **return** $\hat{Y}^c$

Theoretical Analysis

We derive the theoretical performance of our GLDL algorithm (Detailed proof is given in the extended version available in the GitHub project). To proceed, we make some mild assumptions as follows. First, we consider G as simple, undirected graphs with no loop and the maximum degree of $d-1$. Second, the GCN has in total l layers with the maximum hidden dimension k. Third, the nodal feature vectors are normalized and reside in an ℓ_2-ball of radius B, such that $\|x_i^j\|_2 \leq B$, where x_i^j denotes the i-th node's representation at the j-th layer.

Denoted by $L_{\mathcal{G}}(f_w)$ and $L_{(X,A)}(f_w)$ are the generalization error over a graph distribution $\mathcal{G}$ and the empirical error on the training graph data (X, A), respectively, where $(X, A) \overset{\text{iid}}{\sim} \mathcal{G}$. We define $\phi(\cdot,\cdot)$ the distance metric such that $|\phi(u,p) - \phi(u,q)| \leq (\sqrt{m}+1)\|p-q\|_2,\ \forall u,p,q \in \mathbb{R}^m$. The error terms can be defined on the function f_w as follows.

$$L_{\mathcal{G}}(f_w) = \mathbb{E}_{(X,A)\sim\mathcal{G}}\mathbb{E}_{y_i\sim Y}\phi\left(f_w(X,A)[i], y_i\right),$$

$$L_{(X,A)}(f_w) = \frac{1}{nq}\sum_{i=1}^{n}\sum_{j=1}^{q} f_w(X,A)[i,j]\ln\frac{f_w(X,A)[i,j]}{y_{i,j}}$$

where $f_w(X,A)[i] \in \mathbb{R}^q$ and $y_i \in \mathbb{R}^q$ represent the predicted and ground-truth label distribution of the i-th node, respectively. Denoted by $f_w(X,A)[i,j]$ the predicted probability that node i belongs to the j-th class. We then have

Theorem 1 *For any $B > 0, l > 1$, let $f_w \in \mathcal{H} : \mathcal{X} \times \mathcal{G} \to \mathbb{R}^q$ be an l-layer GCN, parameterized by $W_1, \ldots, W_l$. Then for any $\delta, \gamma > 0$, with probability at least $1-\delta$ we have*

$$\begin{aligned} L_{\mathcal{G}}(f_w) - L_{(X,A)}(f_w) \leq\ & \frac{2(\sqrt{2q}+\sqrt{2})q}{\sqrt{n}} \max_{i\in[n],j\in[l]} \left\|\mathbf{x}_i^j\right\|_2 \\ & + 3b\sqrt{\frac{\log 2/\delta}{2n}} \\ & + \mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 k \log(lk)\mathcal{D}(W_i) + \log\frac{nl}{\delta}}{\gamma^2 n}}\right), \end{aligned} \tag{9}$$

where $\mathcal{D}(W_i) = \prod_{i=1}^{l}\|W_i\|_2^2 \cdot \sum_{i=1}^{l}\left(\|W_i\|_F^2 / \|W_i\|_2^2\right)$ bounds the hypothesis space and b is a constant.

Remark Theorem 1 establishes determinants of the generalization error bound of the GLDL algorithm. First, a direct relation between the dimension of the label space q and the generalization error is observed. Specifically, as q increases, the algorithm exhibits less favorable generalization properties. This suggests that the vastness of the label space introduces complexities that adversely impact the generalization capability. Second, our results illuminate that the algorithmic robustness to noise diminishes with the growth of B, which captures the magnitude of data perturbations. Larger values of B indicate heightened numerical instability in the data, which translates to inferior generalization performance. Third, as the graph becomes more intricate (larger d), the algorithm generalizes worse. This emphasizes the challenge of modeling more complex relational data. Fourth, delving into the GCN architecture, we discern that both the depth l and width k of its layers play pivotal roles. A deeper

(larger l) or wider (larger k) GCN tends to exacerbate the generalization error, highlighting the trade-offs inherent in architectural design. Fifth, Upon close examination of the RHS of Eq. (9), we identify structural similarities with the Rademacher complexity of a multi-class γ-margin loss, as suggested by (Kakade, Sridharan, and Tewari 2008), given by $\mathcal{R}(f_w) \leq \max_{i\in[n]} \|x_i\|_2/\sqrt{n}$. It can be succinctly deduced that $\mathcal{R}(f_w, \phi) \leq (\sqrt{2q}+\sqrt{2}) \sum_{j=1}^{c} \mathcal{R}(f_w)$. This reveals that our GLDL algorithm presents a more refined complexity bound compared to a naive decomposition of the label distribution problem into multiple binary regression tasks on graph data. The relaxing coefficient, $\sqrt{2q}+\sqrt{2}$, underscores that GLDL's superiority grows more pronounced with diminishing label vector dimensions.

Complexity Analysis

Asymptotically, GLDL has the complexity of $\mathcal{O}(T_1 qmE + mq^2)$ for static and $\mathcal{O}(T_1(qmE + mq^2) + qm(T_1T_2/\omega))$ for dynamic variants, where q, m, and E denote the numbers of classes, nodal features, and edges, respectively. T_1 and T_2 represent numbers of training epochs for GCN and LLN, respectively, and ω is the frequency of LLN updates. Given that q is much less than m, and E often dominates in large graphs, the term $\mathcal{O}(mq^2)$ becomes relatively trivial; hence, the computing efficiency of our GLDL is on par with vanilla GCNs (Kipf and Welling 2017), which computes at $\mathcal{O}(T_1 qmE)$.

Experiments

Benchmark Datasets

To the best of our knowledge, no public graph dataset with benchmark label distribution is available for evaluation. To verify our model performance, we create four datasets with ground-truth label distributions for each node. Table 1 summarizes the data characteristics. Tables 2 and 3 list average label distributions of all nodes in DBLP and Yelp datasets. Figure 3 further shows average label distributions for one class of Yelp and DBLP dataset, respectively, where Yelp has more even node label distributions whereas DBLP shows much stronger label dominance.

By utilizing heterogeneous datasets and metapath aggregation, homogenization of heterogeneous datasets can be achieved (Fu et al. 2020). During the homogenization process, metapath aggregation allows auxiliary node types or preexisting labels to be converted into distribution labels.

- **DBLP**: Originally a citation network (Tang et al. 2008) composed of Author, Paper, Conference, and Term nodes, DBLP is homogenized through an Author-Paper-Author metapath, resulting in a coauthorship graph where nodes represent authors and each edge indicates that two authors share at least one paper. Each author is labeled with a distribution of conference areas, derived from the author's conference publication history. More specifically, a distribution composed of the conference area in which papers are published, where the distribution is defined as the percentage of papers published at the respective conference area concerning the total number of papers published by the author. The node features are a bag of words representing all authors' papers.
- **Yelp**: Originally a review network composed of Business and User nodes connected by Review, Check-in, and Tip edges, Yelp was aggregated along the Business-User-Business metapath, using review edges, resulting in a network of businesses with common customer bases. The businesses were then labeled with a distribution built from the star ratings extracted from their review edges. Business features are a bag-of-word representation extracted from their reviews.
- **Yelp2**: Another graph generated from the same heterogeneous Yelp graph with a larger bag-of-word feature space. We use Yelp2 to validate how node features impact the algorithm performance for networks with similar topologies.
- **ACM**: Originally a citation network (Tang et al. 2008) composed of Author, Paper, and Subject nodes, ACM was homogenized similarly to DBLP but was labeled using subject nodes instead of the conference nodes. The resultant distribution represents the disciplines in which each author has published. The node features are also a bag-of-word representation of all authors' papers.

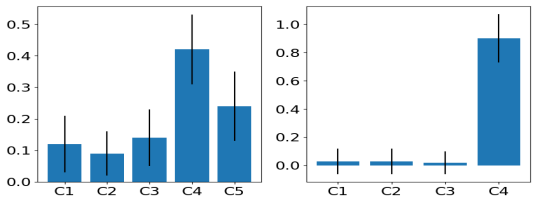

Figure 3: Average label distributions (mean±std) for the 4^{th} class for Yelp (left panel) and the DBLP (right panel) datasets. The nodes whose dominant label is the 4^{th} class are used to calculate the average label distributions.

Baselines

- **SA-IIS** (Geng 2016) SA-IIS uses maximum entropy to find a model minimizing the KL-divergence. When applying SA-IIS to graphs, only node features are used as input, because the model does not consider graph topology. The parametric model can be described as:

$$p(y_i|x;\theta) = \frac{1}{Z} \mathrm{e}^{\sum_k \theta_{y_i,k} x_{i,k}} \tag{10}$$

where $Z = \sum_{y_i \in Y} \mathrm{e}^{\sum_k \theta_{i,k} x_{i,k}}$, $\theta_{i,k}$ is the learnable weight, $x_{i,k}$ is the k-th feature of node v_i. SA-IIS and SA-BGFS have similar performance but SA-BGFS is more efficient in general. SA-IIS is used in our experiments because SA-BGFS frequently encounters stability problem.
- **GCN-KL** A supervised simple Graph Convolution network with multiple GCN layers defined by Eq. (1) and a softmax layer outputs final label distribution for each node. KL-divergence is used as the training loss.

Dataset	Nodes	Edges	Average Degree	Homophily	# of Labels	# of Features	Metapath
DBLP	1711	5796	3.387	0.780	4	334	Author-Paper-Author
Yelp	2719	38233	14.061	0.502	5	1640	Business-User-Business
Yelp2	3000	33857	11.286	0.460	5	6167	Business-User-Business
ACM	6007	25338	4.218	0.920	11	1903	Author-Paper-Author

Table 1: Dataset statistics. Homophily is computed as the fraction of edges connecting nodes sharing the same labels (Ma et al. 2022). Low homophily scores imply heterophilous graphs.

Class 1	Class 2	Class 3	Class 4
$0.88_{\pm 0.18}$	$0.04_{\pm 0.11}$	$0.01_{\pm 0.06}$	$0.07_{\pm 0.12}$
$0.06_{\pm 0.12}$	$0.77_{\pm 0.22}$	$0.08_{\pm 0.15}$	$0.09_{\pm 0.15}$
$0.02_{\pm 0.06}$	$0.04_{\pm 0.10}$	$0.87_{\pm 0.19}$	$0.07_{\pm 0.15}$
$0.03_{\pm 0.09}$	$0.04_{\pm 0.09}$	$0.03_{\pm 0.08}$	$0.99_{\pm 0.17}$

Table 2: DBLP dataset average label distributions (mean±Std) of all nodes *w.r.t.* different classes. To generate this average, label distributions are grouped by their dominant class. Average label distributions are then calculated within their respective groups. Standard deviations are calculated between each class within each group. The table is $q \times q$, where q denotes the number of classes. The diagonal values denote the dominant class's probability value. The lower the diagonal values, the more spread out the class probability is.

Class 1	Class 2	Class 3	Class 4	Class 5
$0.51_{\pm 0.17}$	$0.10_{\pm 0.09}$	$0.09_{\pm 0.09}$	$0.11_{\pm 0.09}$	$0.19_{\pm 0.12}$
$0.13_{\pm 0.11}$	$0.36_{\pm 0.08}$	$0.15_{\pm 0.10}$	$0.20_{\pm 0.11}$	$0.16_{\pm 0.13}$
$0.12_{\pm 0.08}$	$0.12_{\pm 0.10}$	$0.40_{\pm 0.11}$	$0.22_{\pm 0.10}$	$0.14_{\pm 0.11}$
$0.12_{\pm 0.09}$	$0.09_{\pm 0.07}$	$0.14_{\pm 0.09}$	$0.42_{\pm 0.11}$	$0.24_{\pm 0.11}$
$0.11_{\pm 0.10}$	$0.05_{\pm 0.06}$	$0.06_{\pm 0.06}$	$0.13_{\pm 0.11}$	$0.65_{\pm 0.20}$

Table 3: Yelp dataset average label distributions (mean±Std) of all nodes *w.r.t.* different classes (other details are the same as Table 2).

- **GAT-KL** A supervised simple Graph Attention network with multiple GAT layers followed by (Veličković et al. 2018) and a softmax layer outputs final label distribution for each node. KL-divergence is used as the supervised training loss.

Evaluation Metrics

Following commonly used LDL evaluation metrics (Geng 2016), six measures are chosen as our measure for distribution error. Additionally, Weighted F1-score and accuracy on a converted single-label classification setting are also used. These metrics belong to different measure families and each of them reflects some aspects of the model performance.

From our over-smoothing experiments (reported in extended version available in the GitHub project), it is observed that the above six measures may not reflect overall model performance for imbalanced datasets, which typically result in better distribution measure but worse weighted F1-score and accuracy. The addition of the two single-label classification metrics reveals the biased prediction over the dominant class. "# of top metrics" is used to count the number of times a model archives the best or 2^{nd} best performance across all measures.

Results and Analysis

The results in Table 4 show that `GLDL` dynamic model has the best performance with the highest number of winning counts among eight metrics across all the datasets, followed by the static `GLDL` model. SA-IIS has the worst performance. Specifically, SA-IIS only has good accuracy and F1-score on Yelp and Yelp2 datasets but all other distribution measures are significantly lower. One possible reason is that Yelp and Yelp2 have low homophily scores which indicate that they are heterophilic and GCN is known to favor homophilic graphs in a semi-supervised node classification task (Zhu et al. 2020).

GCN-KL and GAT-KL have a better distribution measure than SA-IIS but their overall performance is worse than `GLDL`. This is mainly because they overlook label-label correlation in predicting node label distributions. For `GLDL` static *vs.* dynamic models, the dynamic approach always has a better Chebyshev distance and Intersection and a better weighted F1 score and accuracy among three out of four datasets. The specific experiment settings and further analysis are provided in extended version available in the GitHub project.

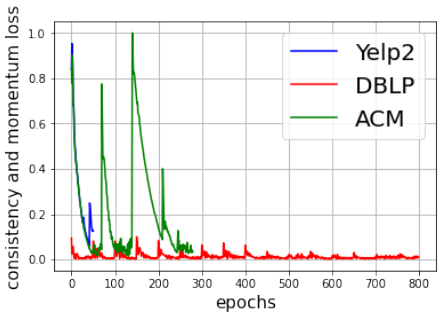

Figure 4: Convergence of the dynamic graph generation *w.r.t* the unsupervised loss. As the number of epochs increases, the loss decreases. The spike denotes the change of the label-label network as per dynamic graph generation. The change of the network results in a larger loss and will decrease after several iterations.

DBLP									
Model	CHD↓	COD↓	CAD↓	CLD↓	IND↑	KLD↓	ACC↑	weighted F1↑	# of top metrics↑
SA-IIS	0.3491	0.2187	3.0655	1.6863	0.6458	0.8321	0.6887	0.6956	0/8
GAT-KL	0.2232	0.1122	**2.8416**	**1.608**	0.7718	0.4037	0.8075	0.7969	2/8
GCN-KL	0.2267	0.0986	2.8706	1.6219	0.7695	**0.3428**	0.8326	0.8321	1/8
GLDL$_s$	*0.2143*	**0.097**	2.8588	*1.6206*	*0.7821*	0.3455	**0.8405**	**0.8406**	6/8
GLDL$_d$	**0.2142**	*0.0977*	*2.8587*	1.6208	**0.7828**	*0.3449*	*0.8327*	*0.8328*	7/8
Yelp									
Model	CHD↓	COD↓	CAD↓	CLD↓	IND↑	KLD↓	ACC↑	weighted F1↑	# of top metrics↑
SA-IIS	0.3682	0.2564	2.7356	1.3972	0.5913	0.8302	0.5196	**0.5655**	1/8
GAT-KL	0.3317	0.2093	2.4005	1.2299	0.6217	0.4685	0.5993	0.4491	0/8
GCN-KL	0.3288	0.1953	2.428	1.2635	0.6311	0.4611	**0.6133**	*0.5296*	2/8
GLDL$_s$	*0.3027*	*0.1824*	*2.3091*	*1.2016*	*0.6550*	*0.4167*	0.5944	0.5035	6/8
GLDL$_d$	**0.2893**	**0.1684**	**2.2432**	**1.1752**	**0.6698**	**0.3843**	*0.6115*	0.5194	7/8
Yelp2									
Model	CHD↓	COD↓	CAD↓	CLD↓	IND↑	KLD↓	ACC↑	weighted F1↑	# of top metrics↑
SA-IIS	0.3628	0.2362	2.8063	1.4299	0.6053	0.8081	0.5196	**0.5655**	1/8
GAT-KL	0.3021	0.1845	2.465	1.2831	0.659	0.4214	0.61	0.5033	0/8
GCN-KL	0.3236	0.1983	2.467	1.2765	0.6352	0.4573	0.57	0.5246	0/8
GLDL$_s$	*0.2957*	*0.1695*	*2.3291*	*1.217*	*0.6682*	*0.3851*	*0.5889*	0.5107	7/8
GLDL$_d$	**0.2874**	**0.1657**	**2.3173**	**1.213**	**0.6741**	**0.3805**	**0.6022**	*0.5456*	8/8
ACM									
Model	CHD↓	COD↓	CAD↓	CLD↓	IND↑	KLD↓	ACC↑	weighted F1↑	# of top metrics↑
SA-IIS	0.42477	0.26459	N/A	N/A	0.57304	1.3635	0.6705	0.6747	0/8
GAT-KL	0.3849	**0.2181**	10.2209	3.1732	0.6108	0.9249	*0.7243*	**0.7052**	3/8
GCN-KL	0.3993	0.2275	10.2388	3.1752	0.5966	0.9247	0.7166	0.691	0/8
GLDL$_s$	*0.3543*	*0.2236*	*9.7422*	*3.0416*	*0.6403*	**0.9093**	0.7116	0.6845	6/8
GLDL$_d$	**0.3534**	0.2252	**9.7354**	**3.0387**	**0.6418**	*0.9132*	**0.726**	*0.7008*	7/8

Table 4: Results of different models on the created datasets. The number of top metrics counts the number of best (Bold) or second-best (Italian) results for each model. CHD: Chebyshev Distance, COD: Cosine Distance, CAD: Canberra Distance, CLD: Clark Distance, IND: Intersection Distance, KLD: KL Divergence, ACC: Accuracy. A ↑ / ↓ symbol denotes that the measured values are higher/lower the better. N/A in the ACM dataset is due to a numerical error from SA-IIS.

Regularization Convergence

Fig. 4 reports unsupervised loss for the graph generator in the training stage (DBLP, Yelp2, and ACM datasets). We can observe that the graph generator $GG(\cdot)$ does achieve a lower consistency and momentum loss during training. The curves in Fig 4 further validate the effectiveness of the GLDL's dynamic module, where all three datasets show clear convergence under dynamic network setting.

Noticeably, ACM has eleven classes (labels), so its label-label network has more variety in topology, making it ideal for assessing dynamic module's performance. As soon as the dynamic network changes, the loss will first experience increase and then decrease. The peak values will gradually decline and eventually converge to a small value.

Conclusion

In this study, we proposed a framework for a novel and under-researched problem: graph label distribution learning, where our goal is to learn from graph structured data to accurately predict label description degrees of respective nodes. We argued that existing LDL methods cannot handle networked data and a simple adaption of GCN using KL-divergence loss is ineffective to solve the problem due to over-smoothing and the lack of consideration of label-label correlation. To address the problem, we proposed GLDL to explicitly model label-label correlation using a unique dynamic graph generation process with unsupervised loss for stable performance. A dual graph convolution network (GCN) co-training with heterogeneous message passing is proposed for node label distribution prediction to fully leverage network topology and label-correlation for accurate prediction. A theoretical bound for GCN training with KL-divergence is derived to support our design. Experimental studies on four benchmark datasets validate GLDL's performance compared to baseline.

Acknowledgments

This work has been supported in part by the National Science Foundation (NSF) under Grant Nos. IIS-2236579, IIS-2302786, IIS-2245946, and IIS-2236578, and in part by the Commonwealth Cyber Initiative (CCI).

References

Bacciu, D.; Errica, F.; Micheli, A.; and Podda, M. 2020. A gentle introduction to deep learning for graphs. *Neural Networks*, 129: 203–221.

Borchani, H.; Varando, G.; Bielza, C.; and Larranaga, P. 2015. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5.

Carbonell, J. G.; Michalski, R. S.; and Mitchell, T. M. 1983. An overview of machine learning. *Machine learning*, 3–23.

Chen, S.; Wang, J.; Chen, Y.; Shi, Z.; Geng, X.; and Rui, Y. 2020. Label distribution learning on auxiliary label space graphs for facial expression recognition. In *CVPR*, 13984–13993.

Chen, Z.-M.; Wei, X.-S.; Wang, P.; and Guo, Y. 2019. Multi-label image recognition with graph convolutional networks. In *CVPR*, 5177–5186.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*, 2331–2341.

Geng, X. 2016. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7): 1734–1748.

Geng, X.; Yin, C.; and Zhou, Z.-H. 2013. Facial age estimation by learning from label distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10): 2401–2412.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 1025–1035.

Jia, X.; Li, W.; Liu, J.; et al. 2018. Label Distribution Learning by Exploiting Label Correlations. In *AAAI*.

Jin, W.; Zhao, L.; Zhang, S.; Liu, Y.; Tang, J.; and Shah, N. 2022. Graph Condensation for Graph Neural Networks. In *ICLR*.

Kakade, S. M.; Sridharan, K.; and Tewari, A. 2008. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *NeurIPS*, 21.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. In *Proc. of the NeurIPS Bayesian Deep Learning Workshop*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Kotsiantis, S. B.; Zaharakis, I. D.; and Pintelas, P. E. 2006. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26: 159–190.

Kou, Z.; Wang, J.; Jia, Y.; and Geng, X. 2023. Exploiting Multi-Label Correlation in Label Distribution Learning. In *ArXiv:2308.01742*.

Ma, Y.; Liu, X.; Shah, N.; and Tang, J. 2022. Is Homophily a Necessity for Graph Neural Networks? In *ICLR*.

Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; and Su, Z. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In *KDD*, 990–998.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Wang, J.; and Geng, X. 2023. Label Distribution Learning by Exploiting Label Distribution Manifold. *IEEE Transactions on Neural Networks and Learning Systems*, 839–852.

Xie, X.; Tian, M.; Luo, G.; et al. 2023. Active learning in multi-label image classification with graph convolutional network embedding. *Future Generation Computer Systems*, 148: 56–65.

Xu, H.; Liu, X.; Zhao, Q.; Ma, Y.; Yan, C.; and Dai, F. 2023. Gaussian Label Distribution Learning for Spherical Image Object Detection. In *CVPR*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Zhang, M.-L.; Zhang, Q.-W.; Fang, J.-P.; Li, Y.-K.; and Geng, X. 2021. Leveraging Implicit Relative Labeling-Importance Information for Effective Multi-Label Learning. *IEEE Transactions on Knowledge and Data Engineering*, 33(5): 2057–2070.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *NeurIPS*, 33: 7793–7804.