Discrete Structure Augmentation for Graph Convolutional Networks

Jianxin Ren, Weining Wu*

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
{jianxin_ren, wuweining}@hrbeu.edu.cn

Abstract

Graph Neural Networks (GNNs) have achieved significant progress in semi-supervised data classification, with an assumption that a complete graph or accurate structure is available. In this paper, a novel GNN architecture, named discrete-structure-augmentation graph convolutional network (DSA-GCN) is proposed, to apply the GCNs in real-world scenarios where the graphs are noisy and incomplete or even not available. Compared with existing methods, DSA-GCN firstly uses a variational Expectation-Maximization (EM) algorithm to jointly learn graph structure, including a discrete probability distribution on the edges of the graph and label dependency, and the parameters of GCN. Second, DSA-GCN applies novel reconstruction loss in learning discrete dependency structure on graph, together with consistency loss. Third, augmentation strategy is used to derive discrete graph structures with varying sparsity. Extensive experiments demonstrate that DSA-GCN significantly outperforms existing methods under varying levels of edge sparsity.

Introduction

As a natural way to represent the relational information, graphs can model the dependency explicitly between data points, and then leverage both attributes of data points and their relationships to train a model of deep learning. Graph Neural Networks (GNNs) (Kipf and Welling 2017) are one such class of algorithms that are able to incorporate sparse and discrete dependency structures in graph, performing well in semi-supervised data classification (Taskar et al. 2007). However, it usually assumes that a complete and reliable graph is available, which is often voilated in practice by noise, missing data, or privacy constraints (Shokri and Shmatikov 2015; Hamilton, Ying, and Leskovec 2017). In order to construct or refine the graph topology with incomplete, noisy or entirely missing structure, affinity graphs (Franceschi et al. 2019; Jin et al. 2020; Kalofolias 2016) that use some measures of similarity between data points in graph have been proposed, but its application to discrete graph structure is limited. Then, later approaches, such as (Zhao, Ding, and Fu 2021; Chen, Wu, and Zaki 2020; Wang et al. 2024), attempt to jointly optimize the graph creation in an end-to-end manner. By learning the global or task-specific dependency between data points, bilevel optimization (Liu et al. 2021), sampling strategy for edges (Zou et al. 2023), and entropy priors in the graph (Duan et al. 2024) is also explored to generate the discrete structure. Overall, learning discrete distribution in graph is a challenging problem, due to the high variance in training a model and the difficulty of preserving connectivity under extreme sparsity.

In most scenarios, GNNs studies (Franceschi et al. 2019; Jin et al. 2020) ignore the label dependency in its training procedure, especially for incomplete graph. For example, studies (Li et al. 2020) lack a unified feedback between the predicted labels and the extracted structures in learning the parameters of model, leading to the error accumulation in training loops and suboptimal generalization of GCNs. By capturing the label dependency, (Qu, Bengio, and Tang 2019) use conditional random fields (CRFs) coupled with GNNs to jointly optimize the structure and parameters of model on a complete graph.

In this paper, we use label dependency as complementary information to jointly optimize the structure and model's parameter on a incomplete graph. Here, the discrete distribution of edges and their label dependency are both learned for deriving the graph structure. We propose Discrete-Structure-Augmentation Graph Convolutional Network (DSA-GCN), a novel GNN architecture, which firstly uses a variational Expectation-Maximization (EM) algorithm to jointly perform discrete graph structure learning and label dependency.

Contributions. We summarize our contributions below:

- A novel GNN architecture named DSA-GCN that jointly performs discrete structure learning and label dependency modeling. In learning discrete graph structures, a GCN-based edge predictor that integrates node features with local structural information is designed, significantly improving the prediction accuracy of missing edges. We innovatively incorporate the structure reconstruction loss into label dependency modeling, explicitly capturing complex label correlations on discrete graphs and effectively enhancing label inference.
- To accurately recover missing edges and facilitate label propagation. We propose a structure augmentation strategy incorporating pseudo-label prediction and cross-

*Corresponding author

set connection mechanisms. By adding high-confidence edges between unlabeled and labeled nodes, this strategy enhances graph connectivity and label propagation. Discrete structures with varying sparsity levels are generated by configuring appropriate sampling parameters.

- Both reconstruction loss and consistency loss into structure learning to jointly optimize the graph structure. On the reconstructed discrete graph, a GCN module GCN_θ is used to learn node representations and perform classification. During optimization, the model jointly minimizes the reconstruction loss and the label modeling loss, and adaptively adjusts graph sparsity via a feedback-driven sampling mechanism to better approximate the true underlying structure. This joint optimization of graph reconstruction and label inference, significantly improving both structural recovery and classification performance.
- We systematically evaluate DSA-GCN on diverse homogenous and heterogeneous graph datasets, covering scenarios with varying degrees of edge sparsity, extreme structural degradation, and complete graphs. Results demonstrate that DSA-GCN consistently achieves stable and superior performance across all sparsity levels, significantly outperforming 20 competitive baselines, including structure-dependent models, weakly structure-aware methods, and structure-learning approaches.

Problem Definition

Following (Qu, Bengio, and Tang 2019), we define a graph $G = (V, E, X_V)$, where V is the node set, $E \subseteq V \times V$ is the edge set, and $X_V \in \mathbb{R}^{n \times d}$ denotes node features. A subset $V_L \subset V$ is labeled with $\mathbf{y}_L$, and the goal is to infer labels for unlabeled nodes $V_U = V \setminus V_L$. To simulate sparsity, we apply Bernoulli sampling with edge deletion probability p_{del} to the adjacency matrix A, yielding a sparse symmetric matrix $\hat{A}$ and a discrete graph $G = (V, \hat{E}, X_V)$, where $\hat{E} = \{(v_i, v_j) \mid \hat{A}_{ij} = 1\}$. Varying p_{del} controls sparsity levels. $\hat{E}$ denotes the initial discrete edge set used as input, and $\widetilde{E}$ represents the edge set after structure augmentation.

Label Dependency. In scenarios with sparse or incomplete graphs, the assumed conditional independence among node labels significantly impairs inference accuracy and generalization. To address this limitation, we explicitly model label dependencies via a Conditional Random Field (CRF). Formally, the joint label distribution is expressed as:

$$p_\phi(\mathbf{y}_V \mid X_V, \widetilde{E}) = \mathcal{F}_{\mathrm{CRF}}(X_V, \widetilde{E}, \mathbf{y}_V), \tag{1}$$

where $\mathcal{F}_{\mathrm{CRF}}$ denotes label dependency modeling process.

Discrete Graph Structure. This process involves two steps: structure reconstruction and label distribution inference. The reconstruction process aims to recover missing edges from the sparse input $\hat{E}$, generating an optimized edge set $\widetilde{E}$. Based on this structure, we perform label distribution inference, which explicitly incorporates the label dependency information modeled in $\mathcal{F}_{\mathrm{CRF}}$ process. This enhances the robustness of classification. Formally,

$$\widetilde{E} = \mathcal{F}_{\text{struct}}(X_V, \hat{E}, \mathbf{y}_L), \tag{2}$$

$$q_\theta(\mathbf{y}_U \mid X_V, \widetilde{E}) = \mathcal{F}_{\text{infer}}(X_V, \widetilde{E}), \tag{3}$$

where $\mathcal{F}_{\text{struct}}$ and $\mathcal{F}_{\text{infer}}$ denote the discrete graph structure reconstruction and lab inference processes, respectively.

DSA-GCN firstly uses a variational EM process alternating between discrete structure learning (E-step), structure optimization and label dependency modeling (M-step).

Edge Distribution Prediction

To accurately predict distribution of missing edges, we propose a graph convolutional network (GCN)-based edge predictor f_{e} as a key component of structure learning. Unlike traditional similarity metrics that rely solely on node features, our method effectively integrates node attributes and local topological information to infer the distribution of potential edges, providing a solid foundation for graph reconstruction and label propagation. The f_{e} learns node representations that integrate structural and attribute information:

$$H = \mathrm{GCN}(E, X_V), \quad H \in \mathbb{R}^{n \times d}, \tag{4}$$

where $H = [h_1^\top, \ldots, h_n^\top]^\top$ denotes the node embedding matrix, and $h_i \in \mathbb{R}^d$ represents the embedding of node v_i. The probability of an edge between nodes is computed via the sigmoid of the inner product between their embeddings: $p_{ij} = \sigma(h_i^\top h_j)$. This design targets challenges from structural sparsity to improve missing edge prediction accuracy.

Structure Augmentation Strategies

Edge sparsity in discrete graphs severely limits message propagation and weakens supervision for unlabeled nodes in semi-supervised classification. To alleviate this issue, we propose two structure augmentation strategies that enhance connectivity and support robust representation learning.

Both strategies establish edges based on node embeddings generated by the edge predictor. Specifically, the first connects each unlabeled node V_U to embedding-similar labeled nodes V_L to facilitate label propagation, while the second links highly similar node pairs within V_U. These structure augmentations serve as pre-processing steps that enrich structural signals for representation learning and can be applied prior to graph reconstruction or inference.

We demonstrate in the experiments that both strategies significantly improve classification performance, especially under severe edge-missing scenarios, and that linking V_U to V_L achieves stronger gains due to more direct supervision.

Methodology

This section introduces the proposed discrete graph structure learning architecture DSA-GCN. The reconstruction and structure augmentation process of discrete graph is illustrated in Figure 1. Built upon the variational EM framework shown in Figure 2, DSA-GCN enables efficient joint optimization and inference of both graph structure and labels.

Model Label Dependency

This section provides a detailed explanation of the label dependency modeling process $\mathcal{F}_{\mathrm{CRF}}$ on the discrete graph. Specifically, we introduce variational distribution $q_\theta(\mathbf{y}_U \mid$

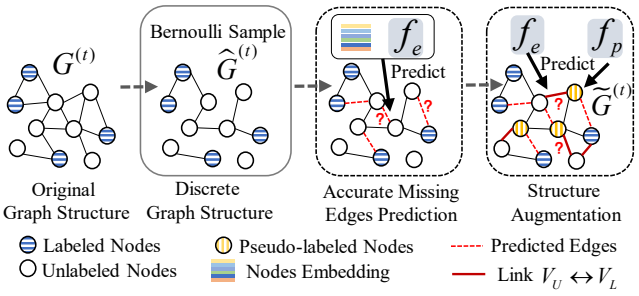

Figure 1: For discrete graph obtained via Bernoulli sampling, a GCN-based edge predictor f_{e} accurately predicts missing edges based on node embeddings. Subsequently, a pseudo-label predictor f_{p} infers labels for V_U, expanding the labeled set. The edge predictor then links V_U to V_L, enhancing the propagation paths of label signals. Red question marks denote low-weight edges removed by thresholding.

$X_V, \widetilde{E}^{(t)})$ to approximate the intractable posterior, explicitly modeling label dependencies using conditional random field (CRF). The joint label distribution is defined as:

$$p_\phi(\mathbf{y}_V \mid X_V, \widetilde{E}^{(t)}) = \frac{1}{Z} \exp\Big(\sum_{n \in V} \psi_u(\mathbf{y}_n, x_n) + \sum_{(i,j) \in \widetilde{E}^{(t)}} \psi_p(\mathbf{y}_i, \mathbf{y}_j, x_i, x_j) \Big), \tag{5}$$

where $\widetilde{E}^{(t)}$ is the reconstructed edge set from E-step at iteration t, and ψ_u and ψ_p denote unary and pairwise potentials.

We optimize q_θ via KL divergence (Kullback and Leibler 1951) minimization to p_ϕ, which is equivalent to maximizing the ELBO (Kingma and Welling 2014). This yields a dependency-aware inference distribution:

$$q_\theta(\mathbf{y}_n \mid X_V, \widetilde{E}^{(t)}) \propto \exp\Big(\psi_u(\mathbf{y}_n, x_n) + \sum_{j \in \mathrm{NB}(n)} \mathbb{E}_{q_\theta}[\psi_p(\mathbf{y}_n, \mathbf{y}_j, x_n, x_j)] \Big). \tag{6}$$

where $\mathrm{NB}(n)$ denotes the neighborhood of node n. This enables unlabeled nodes to leverage high-confidence neighbors for label inference without direct edges, motivating the introduction of label dependency modeling.

In the M-step of the variational EM framework, we instantiate the CRF using a graph neural network GCN_ϕ and apply pseudo-likelihood approximation to capture local label dependencies on the reconstructed graph structure $\widetilde{E}^{(t)}$:

$$\log p_\phi(\mathbf{y}_V \mid X_V, \widetilde{E}^{(t)}) \approx \sum_{n \in V} \log p_\phi\big(\mathbf{y}_n \mid \mathbf{y}_{\mathrm{NB}(n)}, X_V, \widetilde{E}^{(t)}\big), \tag{7}$$

where $\mathbf{y}_{\mathrm{NB}(n)}$ denotes the labels of node n's neighbors.

The conditional label distribution is modeled as:

$$p_\phi\left(\mathbf{y}_n \mid \mathbf{y}_{\mathrm{NB}(n)}, X_V, \widetilde{E}^{(t)}\right) = \mathrm{Cat}\left(\mathbf{y}_n \mid \mathrm{softmax}(W_\phi h_{\phi,n})\right), \tag{8}$$

where $h_{\phi,n} = \mathrm{GCN}_\phi\left(\mathbf{y}_{\mathrm{NB}(n)}, \widetilde{E}^{(t)}\right)$ is the structure-aware representation of node v_n, explicitly capturing label dependencies through the inclusion of $\mathbf{y}_{\mathrm{NB}(n)}$ as feature inputs.

Since ground-truth labels are only available for labeled nodes, we construct training labels as follows:

$$\hat{\mathbf{y}}_n = \begin{cases} \mathbf{y}_n, & v_n \in V_L \\ \text{sample from } q_\theta(\mathbf{y}_n \mid X_V, \widetilde{E}^{(t)}), & v_n \in V_U \end{cases} \tag{9}$$

Based on the pseudo-likelihood approximation, the label modeling loss is defined as the node-wise negative log-likelihood:

$$\mathcal{L}_{\mathrm{label}} = -\sum_{n \in V} \log p_\phi\left(\hat{\mathbf{y}}_n \mid \mathbf{y}_{\mathrm{NB}(n)}, X_V, \widetilde{E}^{(t)}\right). \tag{10}$$

We model label dependency on structure-augmented discrete graph $\widetilde{G}^{(t)}$. To further enhance the effectiveness of this modeling and promote structural optimization, we introduce a graph reconstruction loss $\mathcal{L}_{\mathrm{rec}}$ (Kipf and Welling 2016) as an auxiliary constraint. M-step objective jointly optimizes label modeling and structure refinement:

$$\mathcal{L}_{\mathrm{M}} = \mathcal{L}_{\mathrm{label}} + \lambda \mathcal{L}_{\mathrm{rec}}, \tag{11}$$

where λ balances the two components. This objective enables the firstly effective and feasible modeling of label dependencies on discrete structures. Experiments demonstrate the effectiveness of incorporating the $\mathcal{L}_{\mathrm{rec}}$ into dependency modeling in enhancing overall model performance.

Learn Discrete Structure

Graph Reconstruction. This section provides a detailed description of the reconstruction process $\mathcal{F}_{\text{struct}}$. First, we define a trainable edge probability matrix Θ, where sampled edges follow $A_{ij} \sim \mathrm{Ber}(\Theta_{ij})$. At each E-step, we sample a discrete graph $G^{(t)} = (V, \hat{E}^{(t)}, X_V)$ and identify potential missing edges set $E_{\mathrm{p}}^{(t)}$ based on cosine similarity:

$$E_{\mathrm{k}}^{(t)} = \bigcup_{v_i \in V} \left\{ (v_i, v_j) \mid v_j \in \mathrm{TopK}(v_i, V_L) \setminus \hat{E}^{(t)} \right\}. \tag{12}$$

We estimate connection probabilities via an edge predictor f_{e}, where each score is computed using learned node embeddings h_i and h_j as $p_{ij} = \sigma(h_i^\top h_j)$. Based on a predefined confidence threshold $\tau \in (0, 1)$, we select high-confidence edges as the predicted edge set at iteration t:

$$E_{\mathrm{p}}^{(t)} = \left\{ (v_i, v_j) \in E_{\mathrm{k}}^{(t)} \mid p_{ij} \geq \tau \right\}. \tag{13}$$

Applying a GCN-based pseudo-label predictor f_{p} to infer unlabeled node labels on the updated structure E^+:

$$\hat{Y}^p = \mathrm{softmax}(f_{\mathrm{p}}(X_V, E^+)). \tag{14}$$

Confidently predicted nodes are added to the labeled set:

$$V_{\mathrm{P}} = \left\{ v_i \in V_U^{(t)} \mid \max_c \hat{\mathbf{y}}_i^p[c] \geq \tau_p \right\}. \tag{15}$$

We then update node sets as:

$$\widetilde{V}_L^{(t)} = V_L^{(t)} \cup V_{\mathrm{P}}, \quad \widetilde{V}_U^{(t)} = V \setminus \widetilde{V}_L^{(t)}. \tag{16}$$

To further improve label propagation, we mine confident cross-set edges between labeled and unlabeled nodes:

$$\widetilde{E}_c = \left\{ (v_i, v_j) \mid v_i \in \widetilde{V}_L^{(t)},\ v_j \in \widetilde{V}_U^{(t)},\ p_{ij} \geq \tau' \right\}. \tag{17}$$

The final augmented edge set is $\widetilde{E}^{(t)} = E^+ \cup \widetilde{E}_c$. The resulting graph $\widetilde{G}^{(t)} = (V,\ \widetilde{E}^{(t)},\ X_V)$ is used to model the conditional label distribution $q_\theta(\mathbf{y}_U \mid X_V, \widetilde{E}^{(t)})$, forming the basis for the next round of graph optimization.

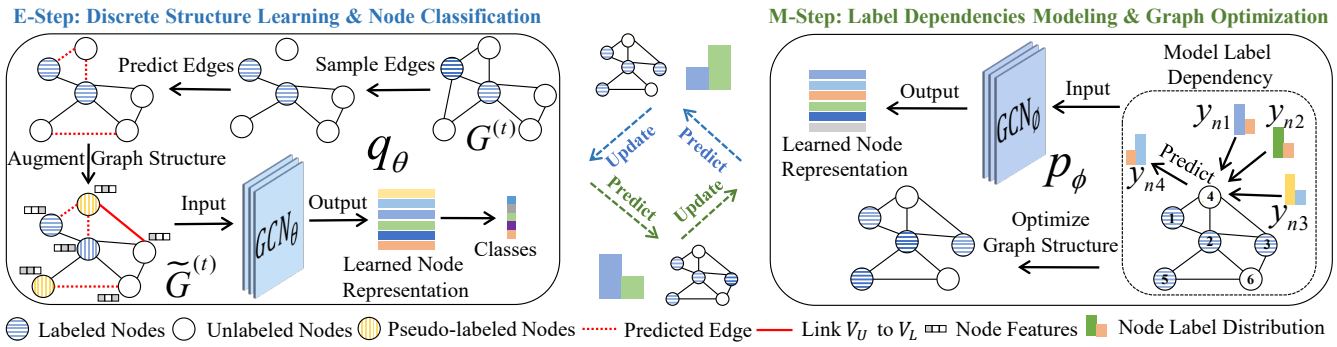

Figure 2: Overview of the proposed DSA-GCN framework, consisting of the E-step for discrete graph structure learning and node classification, and the M-step for label dependency modeling and graph structure optimization.

Label Distribution Inference. Using the label dependency model p_ϕ learned in the M-step, we perform label inference $\mathcal{F}_{\text{infer}}$ and representation learning over the graph $\widetilde{G}^{(t)}$. Following (Kipf and Welling 2016), we adopt a mean-field approximation to factorize the posterior over unlabeled nodes, which is parameterized by a graph convolutional network (GCN) for efficient structure-aware inference:

$$q_\theta(\mathbf{y}_U \mid X_V, \widetilde{E}^{(t)}) = \prod_{v_i \in V} q_\theta(\mathbf{y}_i \mid X_V, \widetilde{E}^{(t)}). \tag{18}$$

Based on this factorization, we construct a conditional inference model $q_\theta(V_U \mid X_V, \widetilde{E}^{(t)})$, which takes node features X_V and the augmented edge set $\widetilde{E}^{(t)}$ as input. The label distribution for each node v_n is modeled as:

$$q_\theta(\mathbf{y}_n \mid X_V, \widetilde{E}^{(t)}) = \text{Cat}\left(\mathbf{y}_n \mid \text{softmax}(W_\theta h_{\theta,n})\right), \tag{19}$$

where $h_{\theta,n} = \text{GCN}_\theta(X_V, \widetilde{E}^{(t)})$ denotes the representation of node v_n, W_θ is a learnable classification weight matrix.

This label inference process serves as the semi-supervised classification objective under the discrete graph. This label inference procedure is inspired by the approach proposed in (Qu, Bengio, and Tang 2019; Li et al. 2019). Supervision comes from labeled nodes, while unlabeled nodes are guided by neighbor-aware pseudo-labels from the M-step model p_ϕ, incorporating label dependencies to enhance learning.

Training Objective. We design a unified multi-task loss that integrates adjacency matrix reconstruction, pseudo-label generation, structure-enhanced consistency, and final classification. This joint training facilitates the mutual improvement of structure completion and label prediction.

Specifically, the edge predictor f_e is optimized via adjacency reconstruction loss $\mathcal{L}_{\text{rec}}$ (Kipf and Welling 2016) with negative sampling to mitigate sparsity-induced bias, thereby enhancing its capability to predict missing edges, the prediction loss is $\mathcal{L}_{\text{p}}$. The pseudo-label predictor f_{p} is supervised using ground-truth labels of currently labeled nodes, and a pseudo-labeled node set is constructed from high-confidence unlabeled nodes. A consistency loss $\mathcal{L}_{\text{add}}$ is further introduced to align predictions between the pseudo-label predictor and the classifier. The classifier GCN_θ is applied to the reconstructed graph for node classification, the loss is $\mathcal{L}_{\text{gcn}}$. We combine all objectives into the final training loss:

$$\mathcal{L}_{\text{E}} = \mathcal{L}_{\text{gcn}} + \mathcal{L}_p + \alpha \cdot \mathcal{L}_{\text{rec}} + \beta \cdot \mathcal{L}_{\text{add}}, \tag{20}$$

where $\alpha, \beta \in \mathbb{R}^+$ are tunable hyperparameters balancing the reconstruction and consistency terms.

Algorithm 1: Training & Optimization Algorithm

Input: Discrete graph $G^{(t)}$, labeled nodes $(V_L, \mathbf{y}_L)$
Output: Optimal reconstructed graph $\widetilde{G}^{(t)}$, $\mathbf{y}_U$ for V_U
Initialize GCN_θ, GCN_ϕ, predictor f_e and f_p
Reconstruct discrete graph $G^{(t)}$ following $\mathcal{F}_{\text{struct}}$
Pre-train q_θ and stabilize learning process
while not converged **do**
  M-Step: Modeling and Optimization
  Compute $\mathcal{L}_{rec}$ using $\widetilde{E}^{(t+1)}$
  Compute total loss $\mathcal{L}_{\text{M}}$ according to Eq. (11)
  Adjust sample ratio δ based on $\mathcal{L}_{\text{M}}$
  Guide the optimization of Θ with δ
  Model label dependencies following $\mathcal{F}_{\text{CRF}}$
  Train using final modeling loss $\mathcal{L}_{\text{M}}$
  E-Step: Learning and Classification
  Sample $\hat{E}^{(t)} \leftarrow \text{SampleGraph}(\widetilde{E}^{(t-1)}, \Theta, \delta)$
  Reconstruct discrete graph following $\mathcal{F}_{\text{struct}}$
  Obtain reconstructed graph $\widetilde{E}^{(t+1)}$
  Obtain updated graph $\widetilde{G}^{(t+1)} = (V, \widetilde{E}^{(t+1)}, X_V)$
  Train using the multi-task loss $\mathcal{L}_{\text{E}}$ in Eq. (20)
end while
Classify each unlabeled node n with $q_\theta(\mathbf{y}_n \mid X_V, \widetilde{E})$

Optimization

We divide the optimization process into two components.

Reconstructed Graph Optimization. In the M-step, besides modeling label dependencies, we optimize the graph generator parameter matrix $\Theta \in [0,1]^{N \times N}$ to guide subsequent structure sampling. We incorporate a reconstruction loss $\mathcal{L}_{\text{rec}}$ into dependencies modeling, which encourages the sampled adjacency matrix to better approximate the original graph, enhancing both learnability and structural fidelity.

To control sampling sparsity, we employ a dynamic adjustment strategy. Specifically, the sampling rate δ is decreased when the loss $\mathcal{L}_{\mathrm{M}}$ drops to enhance stability, and increased otherwise to promote structural exploration. The adjustment is performed with a fixed step size η. The optimized graph structure is subsequently fed back into the E-step as input, facilitating further label inference.

Overall Framework Optimization. Experimentally, the inference model q_θ consistently outperforms the joint model p_ϕ in both accuracy and stability, therefore used for final classification. As illustrated in Figure 2, the E-step reconstructs the discrete graph structure $\widetilde{G}^{(t)}$ via edge prediction and structure augmentation. The inference model q_θ then leverages $\widetilde{G}^{(t)}$ to learn node representations and infer label distributions for unlabeled nodes. In the M-step, the reconstructed graph is first optimized, followed by updating p_ϕ to capture label dependencies based on neighboring pseudo-labels and node features. We implement q_θ and p_ϕ using GCN_θ and GCN_ϕ. The E-step supplies refined structure for the M-step's dependencies modeling, while the M-step feedbacks label distributions to guide confident edge selection in the E-step, enabling closed-loop optimization. The overall training and optimization procedure is detailed in Alg 1.

Computational Complexity

For the GCN-based edge predictor f_e and pseudo-label predictor f_p, the time complexity is $O(Nd^2(L-1)+NdD+|\hat{E}|dL)$, and for the GCN_θ and GCN_ϕ, the complexity is $O(Nd^2(L-1)+NdD+|\tilde{E}|dL)$, where N, L, d, and D denote the number of nodes, GCN layers, input and hidden dimensions, and $|\hat{E}|, |\tilde{E}|$ are the numbers of edges in the discrete and reconstructed graphs. In the missing edge prediction phase, connection probabilities for NK candidate edges are computed via inner products of node embeddings, yielding a time complexity of $O(NKd)$, where K is the candidate neighbor size. During structure augmentation, adding E_c cross-set edges incurs a cost of $\mathcal{O}(|E_c|d)$, which is typically small and negligible confidence filtering. Both loss terms $\mathcal{L}_E$ and $\mathcal{L}_M$ have complexity $O(|\tilde{E}|dL)$. Therefore, the overall time complexity per training iteration of DSA-GCN is $O(Nd^2(L-1)+NdD+|\hat{E}|dL+NKd+|\tilde{E}|dL+ +|E_c|d) \approx O(N+|\hat{E}|+|\tilde{E}|)$. This shows that the time complexity of DSA-GCN grows linearly with the number of nodes N and edges $|\hat{E}|$, $|\tilde{E}|$, ensuring efficiency and scalability. While graph sampling is $O(N^2)$ in the worst case, it can be efficiently parallelized via Bernoulli or matrix sampling. Prior studies (Shin, Shomorony, and Zhao 2023; Qiu et al. 2025) achieved millions of edges sampled per second on OGB-scale graphs, thereby supporting both the theoretical and practical scalability of our approach.

Experiment

We design four experiment objectives to systematically evaluate DSA-GCN: (1) graph learning and edge prediction under incomplete structures, (2) generalization across diverse graph types, (3) robustness under extreme graph sparsity, (4) performance enhancement on complete graphs.

Datasets

We evaluate DSA-GCN on standard citation networks Cora, Citeseer, and Pubmed (Sen et al. 2008). To test robustness under structural degradation, we randomly remove 20%, 40%, 60%, 80%, and 100% of edges, simulating varying levels of sparsity. To evaluate generalization under diverse graph structures, experiments are conducted on additional homogenous graphs (Amazon-Photo, Amazon-Computers, Coauthor-CS, Coauthor-Physics, Wiki-CS) and heterogeneous graphs (ACM, DBLP) (Shchur et al. 2018; Mernyei and Cangea 2020; Wang et al. 2019; Tu et al. 2018). Except for citation datasets, all others use a fixed split of 48% training, 32% validation, and 20% testing (Yang, Cohen, and Salakhudinov 2016) to ensure consistency.

Baselines and Setup

In the edge-missing setting, we compare DSA-GCN with the standard GCN (Kipf and Welling 2017). Within our discrete structure learning framework, we introduce an edge predictor f_{e} that infers missing edges from learned embeddings, instead of relying solely on raw node features. To assess its effectiveness, we design three alternatives: (1) **DSA-feature**, which constructs edges via feature similarity (Jin et al. 2020); (2) **DSA-kNN**, which uses k-nearest neighbors; and (3) **DSA-random**, which adds edges randomly.

To simulate severe sparsity, we randomly drop most edges (breaking connectivity) and compare three categories of baselines. (1) **Structure-dependent GNNs**, like GCN, GAT, GCNII (Chen et al. 2020); (2) **Weakly structure-dependent models**, such as LP, GraphMLP (Hu et al. 2021), GPRGNN (Chien et al. 2021); (3) **Structure-learning GNNs**, including AdaGNN, LSGNN (Chen et al. 2023), MGCN (Yang et al. 2024) and AGMixup-GCN (Lu et al. 2025). For implementation, f_{e}, f_{p}, GCN_ϕ, and GCN_θ are two-layer GCNs with 16 hidden units, ReLU activation, and a final softmax classifier (Kipf and Welling 2017). Dropout (Srivastava et al. 2014) with $p = 0.5$ is applied. Training uses RMSprop (Tieleman and Hinton 2017) or Adam (Kingma and Ba 2015) with learning rates in $\{0.005, 0.01, 0.05\}$ and weight decay of 0.0005. Models is trained for 150 epochs per iteration. Results are averaged over five runs with standard deviations reported.

Datasets	Ratio	Original	$V_U \leftrightarrow V_U$	$V_U \leftrightarrow V_L$
Cora	20%	78.2 ± 0.3	79.4 ± 0.2	80.9 ± 0.3
	40%	75.6 ± 0.4	77.6 ± 0.3	79.7 ± 0.2
	60%	73.6 ± 0.2	75.2 ± 0.7	76.4 ± 0.3
CiteSeer	20%	66.8 ± 0.3	68.6 ± 0.4	70.7 ± 0.5
	40%	65.3 ± 0.7	67.7 ± 0.2	69.9 ± 0.2
	60%	63.4 ± 0.8	65.8 ± 0.5	68.2 ± 0.3

Table 1: Impact of structure augmentation strategies on node classification accuracy under different edge-missing. "Ratio" denotes the proportion of edges removed by Bernoulli sampling, representing the loss rate relative to all edges.

Method	Cora	Pubmed	Photo	Computers	CS	Physics	Wiki-CS	DBLP	ACM
GCN	52.6 ± 0.4	63.1 ± 0.3	81.8 ± 0.6	60.0 ± 0.6	79.0 ± 1.0	82.9 ± 0.4	47.7 ± 0.6	69.8 ± 0.5	75.9 ± 0.8
GAT	64.2 ± 0.5	60.1 ± 0.6	84.4 ± 0.7	70.8 ± 0.4	77.8 ± 0.6	82.0 ± 0.6	23.0 ± 0.7	68.6 ± 0.6	74.9 ± 0.6
GCNII	65.3 ± 0.3	74.2 ± 0.3	24.6 ± 1.2	36.8 ± 0.3	44.3 ± 0.6	79.7 ± 0.2	56.9 ± 0.4	70.6 ± 0.2	74.0 ± 0.4
GNN-LF	72.3 ± 0.3	73.8 ± 0.5	77.3 ± 0.5	60.8 ± 0.3	84.9 ± 0.8	91.3 ± 0.2	24.5 ± 0.1	75.2 ± 0.3	89.6 ± 0.6
LP	57.9 ± 0.2	51.7 ± 0.2	77.9 ± 1.2	75.9 ± 0.6	77.6 ± 0.2	83.2 ± 0.3	60.6 ± 0.2	69.1 ± 0.7	73.5 ± 0.9
SemiEmb	53.9 ± 0.4	55.8 ± 0.8	76.8 ± 0.5	45.2 ± 0.7	68.1 ± 0.9	79.6 ± 0.3	67.7 ± 0.8	67.8 ± 0.4	73.7 ± 0.2
GPRGNN	65.4 ± 3.1	$\mathbf{80.1 \pm 0.6}$	87.6 ± 2.2	77.1 ± 2.8	87.9 ± 0.5	92.9 ± 0.3	73.4 ± 0.9	75.6 ± 0.8	87.6 ± 1.1
GraphMLP	71.1 ± 1.2	74.2 ± 1.1	41.8 ± 3.9	50.2 ± 1.5	43.2 ± 0.9	76.2 ± 1.3	66.8 ± 0.6	59.5 ± 1.4	72.9 ± 0.3
GRAND	74.2 ± 0.8	74.6 ± 1.2	88.4 ± 0.8	76.6 ± 1.2	85.8 ± 0.8	87.6 ± 0.9	66.7 ± 0.3	76.8 ± 0.4	92.6 ± 0.7
AdaGNN	68.2 ± 0.6	72.9 ± 0.6	87.9 ± 1.2	80.8 ± 0.6	$\mathbf{93.2 \pm 0.6}$	87.3 ± 1.2	68.8 ± 1.1	78.1 ± 1.9	70.8 ± 1.2
LSGNN	65.6 ± 0.5	70.8 ± 0.3	$\mathbf{93.1 \pm 0.2}$	86.8 ± 0.2	90.1 ± 0.4	89.3 ± 0.6	77.4 ± 0.3	80.7 ± 0.2	92.2 ± 0.3
MGCN	65.4 ± 0.1	80.9 ± 0.2	88.9 ± 0.6	82.4 ± 0.9	86.7 ± 0.2	88.8 ± 0.3	71.7 ± 0.5	74.9 ± 0.2	85.1 ± 0.3
AGMixup	69.2 ± 0.6	75.8 ± 0.1	83.1 ± 0.2	74.5 ± 0.6	83.6 ± 0.4	86.3 ± 0.1	62.2 ± 0.6	72.2 ± 0.5	90.4 ± 0.7
Ours	$\mathbf{75.4 \pm 0.5}$	$\mathbf{78.4 \pm 0.9}$	$\mathbf{92.8 \pm 0.4}$	$\mathbf{88.8 \pm 0.3}$	$\mathbf{91.3 \pm 0.2}$	$\mathbf{94.9 \pm 0.5}$	$\mathbf{79.1 \pm 0.7}$	$\mathbf{80.9 \pm 0.2}$	$\mathbf{94.3 \pm 0.4}$

Table 2: Node classification accuracy (%) comparison under extremely sparse graph structures.

Data	20%	40%	60%	80%	100%
Cora	83.6 ± 0.3	81.8 ± 0.4	79.2 ± 0.4	78.3 ± 0.2	77.1 ± 0.7
Cite.	73.5 ± 0.3	72.1 ± 0.2	71.7 ± 0.4	71.6 ± 0.4	71.2 ± 0.4
Pub.	80.3 ± 0.2	79.1 ± 0.3	78.3 ± 0.4	77.9 ± 0.4	77.7 ± 0.3
Dblp	84.3 ± 0.3	83.4 ± 0.3	82.2 ± 0.3	81.6 ± 0.1	81.6 ± 0.3
Acm	93.9 ± 0.4	93.8 ± 0.3	93.4 ± 0.3	92.7 ± 0.3	91.7 ± 0.2
Pho.	93.7 ± 0.5	93.8 ± 0.4	93.2 ± 0.4	92.1 ± 0.6	89.5 ± 0.3
Com.	89.2 ± 0.3	89.1 ± 0.2	86.6 ± 0.4	87.0 ± 0.2	84.5 ± 0.4
CS	93.4 ± 0.3	92.8 ± 0.3	91.6 ± 0.3	90.8 ± 0.1	89.5 ± 0.4
Phys.	95.7 ± 0.4	95.5 ± 0.2	95.3 ± 0.4	94.5 ± 0.4	94.1 ± 0.3
Wiki.	80.4 ± 0.2	80.2 ± 0.5	79.3 ± 0.4	77.8 ± 0.5	75.1 ± 0.2

Table 3: DSA-GCN node classification under random edge deletion with structural connectivity maintained.

Results

We first validate the structure augmentation strategies. As shown in Table 1, both enhance classification by improving label propagation, with linking V_U to V_L yielding the largest gains and maintaining robustness under extreme sparsity.

We then systematically evaluate DSA-GCN across objectives. As shown in Figures 3–5, it maintains stable accuracy on Cora and Pubmed under varying edge retention ratios. It consistently outperforms all baselines across different edge construction strategies and converges efficiently, demonstrating strong learning and optimization.

We randomly remove 20%–100% of edges while preserving connectivity. Table 3 shows that DSA-GCN maintains competitive accuracy despite increasing edge sparsity. Even with all edges removed, it achieves strong performance (77.1% on Cora, 91.7% on ACM), demonstrating robustness across both homogenous and heterogeneous graphs.

We drop 60% of edges without preserving connectivity, following (Chamberlain et al. 2021; Franceschi et al. 2019; Luo et al. 2022). As Table 2 shows, DSA-GCN consistently outperforms GCN and structure-learning baselines, achieving 94.3% on ACM and 94.9% on Coauthor-Physics, showing strong robustness under realistic edge-missing scenarios.

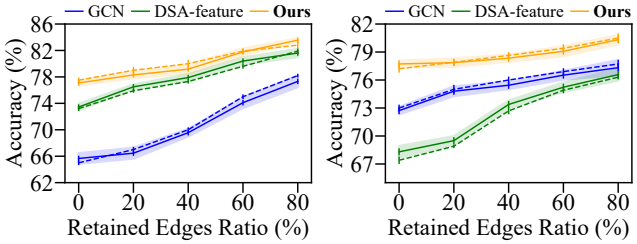

Figure 3: DSA-GCN performance under varying edge sparsity on Cora (left) and Pubmed (right). Solid/dashed lines: test/validation accuracy; shaded areas: standard deviation.

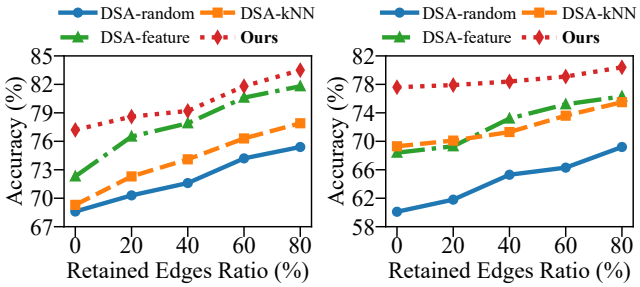

Figure 4: Comparison of four edge reconstruction strategies on Cora (left) and Pubmed (right).

Under standard settings (Table 4), DSA-GCN achieves 84.3%, 74.2%, and 80.5% accuracy on Cora, Citeseer, and Pubmed, outperforming GCN, GAT, DropEdge (Rong et al. 2020), GCNII, APPNP, G^{2}CN (Li et al. 2022), t-hopGCN (Jiao et al. 2023) and AAGCN (Wang et al. 2024), demonstrating its ability to leverage latent structure and label dependencies even on complete graphs.

Clustering Visualization Analysis

To assess representation quality under extreme sparsity, we conduct experiments on Cora (top) and Citeseer (bottom)

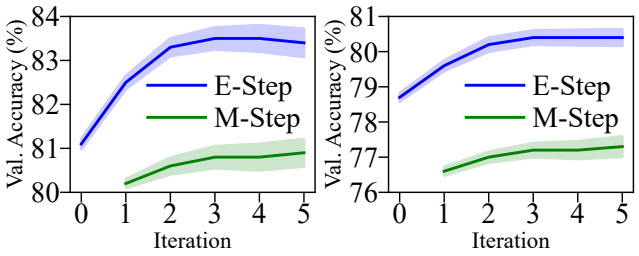

Figure 5: Convergence on Cora (left) and Pubmed (right).

Model or Method	Cora	Citeseer	Pubmed
GCN	81.5 ± 0.5	70.3 ± 0.7	79.0 ± 0.5
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
APPNP	83.8 ± 0.3	71.6 ± 0.5	79.7 ± 0.3
DropEdge(GCN)	82.8	72.3	79.6
GCNII(16 layers)	83.2 ± 0.5	72.1 ± 0.6	80.2 ± 0.4
GNN-LF	83.1 ± 0.2	71.8 ± 0.4	80.0 ± 0.3
GraphMLP	79.5 ± 0.5	73.1 ± 0.2	79.7 ± 0.3
G^2CN	82.7	73.8	80.4
GraphSAP	81.5	71.7	78.9
t-hopGCN	82.5	71.0	79.0
AAGCN	83.3	71.6	80.4
Gseg	82.4	67.4	79.1
Ours	$\mathbf{84.3 \pm 0.2}$	$\mathbf{74.2 \pm 0.2}$	$\mathbf{80.5 \pm 0.4}$

Table 4: Node classification accuracy (%) on complete graphs and DSA-GCN's structural enhancement evaluation.

by removing 40%, 60%, and 80% of edges, along with a complete graph baseline for comparison. Node embeddings learned by DSA-GCN are visualized via t-SNE in Figure 6. Even with 80% edge removal, DSA-GCN produces well-separated clusters with compact intra-class and clear inter-class distributions, indicating strong representation learning.

Ablation Study

To investigate the contribution of each component under severe sparsity (60% edges removed), we conduct an ablation study by removing individual modules. As shown in Table 5, removing graph reconstruction leads to the largest drop (6.08% on Cora, 7.47% on Citeseer), demonstrating its crucial role. Removing the edge predictor or edge augmentation also results in performance degradation, highlighting the benefit of structure refinement. Disabling the variational EM causes a dramatic drop (up to 19.15%), confirming the importance of joint structure optimization and modeling.

Hyperparameter Sensitivity Analysis

We analyze the sensitivity of four key hyperparameters: the edge selection threshold τ, the graph reconstruction loss weight (M-step) α, the consistency regularization coefficient β, and the graph reconstruction loss weight (E-step) λ. Figure 7 shows Cora results. The accuracy (80.2%) is achieved at $\tau = 0.5$, indicating that a moderate threshold balances informative edges and noise suppression. In the M-step, $\alpha = 0.02$ optimally balances original and new structural information. Consistency regularization with $\beta = 0.5$ yields the accuracy (80.5%), facilitating mutual supervision between the pseudo-label predictor and classifier. The E-step weight $\lambda = 0.01$ (79.9%) confirms the benefit of integrating structural loss into label dependency modeling. Proper hyperparameter tuning significantly improves performance.

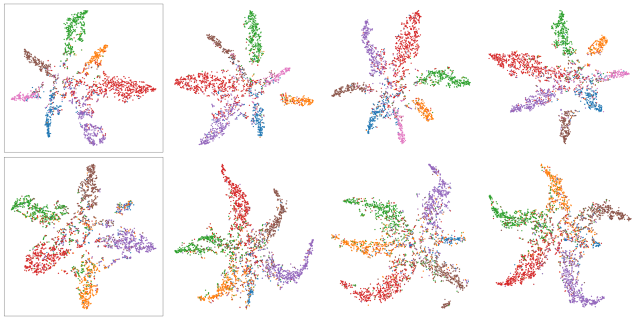

Figure 6: t-SNE of node embeddings on Cora and Citeseer.

Ablation Setting	Cora			Citeseer		
	Acc.(%)	Std.	Drop	Acc.(%)	Std.	Drop
Full DSAGCN	79.18	0.40	–	71.72	0.30	–
w/o Reconstruction	73.10	0.12	-6.08	64.25	0.22	-7.47
w/o Edge Predictor	76.85	0.54	-2.33	67.61	0.50	-4.11
w/o Augmentation	74.62	0.15	-4.56	65.90	0.54	-5.82
w/o EM framework	74.95	0.54	-4.23	52.57	0.40	-19.15
w/o E-Step	77.56	0.28	-1.62	67.60	0.15	-4.12
One-Shot Inference	76.30	0.74	-2.88	67.20	0.30	-4.52

Table 5: Ablation results, each row disables a component: graph reconstruction, edge predictor, unlabeled-to-labeled augmentation, variational EM framework, E-step (modeling and optimization), or one-shot training without iteration.

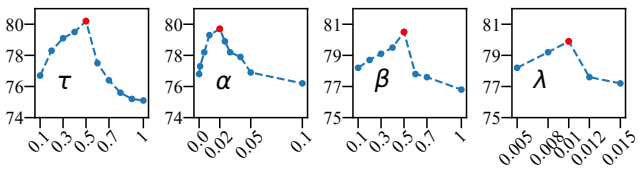

Figure 7: Accuracy on Cora with Varying Hyperparameters.

Conclusion

In this work, we propose a novel GNN architecture that integrates discrete graph structure learning and label dependency modeling, significantly enhancing semi-supervised classification on discrete graphs. First, we theoretically and empirically demonstrate the effectiveness of the proposed edge prediction and structure augmentation strategies. Then, we model label dependencies to alleviate the impact of structural sparsity on label inference and further optimize the reconstructed graph. Extensive experiments conducted on various datasets under different edge sparsity levels show that our method consistently outperforms existing approaches.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 61976067.

References

Chamberlain, B.; Rowbottom, J.; Gorinova, M. I.; Bronstein, M.; Webb, S.; and Rossi, E. 2021. Grand: Graph neural diffusion. In *International conference on machine learning*, 1407–1418. PMLR.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *International conference on machine learning*, 1725–1735. PMLR.

Chen, Y.; Luo, Y.; Tang, J.; Yang, L.; Qiu, S.; Wang, C.; and Cao, X. 2023. LSGNN: Towards General Graph Neural Network in Node Classification by Local Similarity. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, 3550–3558. ijcai.org.

Chen, Y.; Wu, L.; and Zaki, M. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33: 19314–19326.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

Duan, L.; Chen, X.; Liu, W.; Liu, D.; Yue, K.; and Li, A. 2024. Structural entropy based graph structure learning for node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 8372–8379.

Franceschi, L.; Niepert, M.; Pontil, M.; and He, X. 2019. Learning discrete structures for graph neural networks. In *International conference on machine learning*, 1972–1982. PMLR.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 1025–1035. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Hu, Y.; You, H.; Wang, Z.; Wang, Z.; Zhou, E.; and Gao, Y. 2021. Graph-mlp: Node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*.

Jiao, Q.; Zhao, P.; Zhang, H.; Han, Y.; and Liu, G. 2023. Path-enhanced graph convolutional networks for node classification without features. *Plos One*, 18(6): e0287001.

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 66–74.

Kalofolias, V. 2016. How to learn a graph from smooth signals. In *Artificial intelligence and statistics*, 920–929. PMLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *CoRR*, abs/1611.07308.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. Toulon, France.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86.

Li, G.; Muller, M.; Thabet, A.; and Ghanem, B. 2019. DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF international conference on computer vision*, 9267–9276.

Li, M.; Guo, X.; Wang, Y.; Wang, Y.; and Lin, Z. 2022. G ^{2} CN: Graph Gaussian Convolution Networks with Concentrated Graph Filters. In *International Conference on Machine Learning*, 12782–12796. PMLR.

Li, P.; Wang, Y.; Wang, H.; and Leskovec, J. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33: 4465–4478.

Liu, R.; Liu, X.; Yuan, X.; Zeng, S.; and Zhang, J. 2021. A value-function-based interior-point method for non-convex bi-level optimization. In *International conference on machine learning*, 6882–6892. PMLR.

Lu, W.; Guan, Z.; Zhao, W.; Yang, Y.; Zhan, Y.; Lu, Y.; and Tao, D. 2025. Agmixup: Adaptive graph mixup for semi-supervised node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 19143–19151.

Luo, Z.; Lian, J.; Huang, H.; Jin, H.; and Xie, X. 2022. Adagnn: Adapting to local patterns for improving graph neural networks. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 638–647.

Mernyei, P.; and Cangea, C. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*.

Qiu, R.; Jiang, X.; Fang, Y.; Lai, H.; Miao, H.; Chu, X.; Zhao, J.; and Wang, Y. 2025. Efficient Graph Continual Learning via Lightweight Graph Neural Tangent Kernels-based Dataset Distillation. In *Forty-second International Conference on Machine Learning*.

Qu, M.; Bengio, Y.; and Tang, J. 2019. Gmnn: Graph markov neural networks. In *International conference on machine learning*, 5241–5250. PMLR.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations (ICLR)*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop, NeurIPS*.

Shin, S.; Shomorony, I.; and Zhao, H. 2023. Efficient learning of linear graph neural networks via node subsampling. *Advances in Neural Information Processing Systems*, 36: 55479–55501.

Shokri, R.; and Shmatikov, V. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 1310–1321.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958.

Taskar, B.; Abbeel, P.; Wong, M.-F.; and Koller, D. 2007. Relational markov networks. *Introduction to statistical relational learning*, 175: 200.

Tieleman, T.; and Hinton, G. 2017. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *Technical report*.

Tu, K.; Cui, P.; Wang, X.; Yu, P. S.; and Zhu, W. 2018. Deep recursive network embedding with regular equivalence. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2357–2366.

Wang, B.; Cai, B.; Sheng, J.; and Jiao, W. 2024. AAGCN: a graph convolutional neural network with adaptive feature and topology learning. *Scientific Reports*, 14(1): 10134.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The world wide web conference*, 2022–2032.

Yang, X.; Wang, Y.; Liu, Y.; Wen, Y.; Meng, L.; Zhou, S.; Liu, X.; and Zhu, E. 2024. Mixed graph contrastive network for semi-supervised node classification. *ACM Transactions on Knowledge Discovery from Data*, 18(7): 1–19.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 40–48. PMLR.

Zhao, K.; Ding, K.; and Fu, Y. 2021. Learning optimal graph structure via neural generation model. In *International Conference on Learning Representations (ICLR)*.

Zou, X.; Li, K.; Chen, C.; Yang, X.; Wei, W.; and Li, K. 2023. DGSLN: Differentiable graph structure learning neural network for robust graph representations. *Information Sciences*, 626: 94–113.