# An Efficient Prototype-Based Clustering Approach for Edge Pruning in Graph Neural Networks to Battle Over-Smoothing

**Yuyang Huang**[1], **Wenjing Lu**[1] and **Yang Yang**[1,2,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai, China
{yuyanghuang1999, luluerji}@sjtu.edu.cn, yangyang@cs.sjtu.edu.cn

## Abstract

Topology augmentation is a popular strategy to address the issue of over-smoothing in graph neural networks (GNNs). To prevent potential distortion of node representations, an essential principle is to enhance the separability between embeddings of nodes from different classes while preserving smoothness among nodes of the same class. However, differentiating between inter-class and intra-class edges becomes arduous when class labels are unavailable or the graph is partially labeled. While clustering offers an alternative for identifying closely connected groups of nodes, traditional clustering methods face challenges when applied to GNNs in terms of accuracy, efficiency, adaptability, and scalability to diverse graphs. To address these limitations, we introduce ClusterDrop, which uses learnable prototypes for efficient clustering and incorporates supervised signals to enhance accuracy and adaptability across different graphs. Experiments on six datasets with varying graph structures demonstrate its effectiveness in alleviating over-smoothing and enhancing GNN performance.

## 1 Introduction

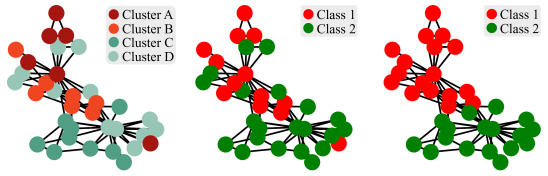
Graph neural networks (GNNs) have emerged as a powerful approach for learning from graph-structured data, and have demonstrated remarkable efficacy across diverse applications [Kipf *et al.*, 2018; Ding *et al.*, 2019; Zheng *et al.*, 2020]. Nevertheless, in contrast to other deep neural network architectures, GNNs tend to be shallow due to the prevalent challenge of over-smoothing. Over-smoothing is a phenomenon where node embeddings collapse into a subspace that loses too much information, causing the embeddings to become too similar to be distinguished in downstream tasks such as node classification.

The issue of over-smoothing was initially introduced by [Li *et al.*, 2018] and has since received wide attention within the machine learning community. Previous works on alleviating the over-smoothing issue can be roughly divided into three types, namely mechanism modification, embedding regularization, and topology augmentation. Some works attempt to alleviate over-smoothing by designing new message-passing mechanisms based on graph theory [Xu *et al.*, 2018; Gasteiger *et al.*, 2019; Chen *et al.*, 2020b; Wu *et al.*, 2023a; Jiang *et al.*, 2022], while most of the designs are at the whole-model level, thus limiting their scalability for different types of GNNs. The studies of embedding regularization try to relieve the excessive similarity of node embeddings by performing feature-level regularization [Zhao and Akoglu, 2020; Guo *et al.*, 2023]. A drawback of these methods is the risk of undermining the inherent smoothness of GNNs, which has been shown to be crucial to the performance of GNNs [Chen *et al.*, 2020a; Wang *et al.*, 2022; Keriven, 2022]. In recent years, topology augmentation through edge addition and removal has gained popularity in addressing the over-smoothing issue [Rong *et al.*, 2020; Chen *et al.*, 2020a; Zhao *et al.*, 2021; Wang *et al.*, 2022; Liu *et al.*, 2023]. The primary concept behind edge modification is to retain intra-class edges while dropping inter-class edges, where a key insight involving two aspects can be summarized: avoiding smoothness between node groups of different classes while preserving smoothness within groups of same-class nodes. However, accurately discerning inter-class edges poses a significant challenge in cases where class labels are unavailable or the graph is partially labeled. Previous attempts set criteria for edge addition or removal based on prior assumptions, while they do not adapt to different graphs. Another viable approach involves training an edge predictor to guide edge modification, but sampling weights for all edges to perform end-to-end training is memory inefficient. Therefore, there is a need to explore more effective methods for augmenting graph topology.

The topological structures of many graphs exhibit modularity naturally [Newman, 2006]. Modularity provides potent prior knowledge, offering valuable cues regarding node properties, including their respective classes. This effect is illustrated using Zachary's Karate Club graph [Zachary, 1977] as shown in Fig. 1. To extract node features, a one-layer untrained GCN is employed on the graph. Then we perform $k$-means clustering to group the nodes into 4 clusters based on the extracted features. We find that the clustering results capture the modularity (Fig. 1(a)), and substantially reflect the classes of the nodes (Fig. 1(b) and (c)). Furthermore, the node clusters naturally partition the graph into distinct groups

*Corresponding author

and offer guidance for edge dropping between different class nodes, which aligns well with the insight mentioned before. This inspiration prompts us to employ clustering to address the over-smoothing issue.




(a) Clustering results (b) Predicted labels (c) Ground truth

Figure 1: An illustration for capturing modularity by $k$-means. The predicted labels in (b) are obtained by assigning each cluster to a class through majority voting.

However, applying traditional clustering algorithms to GNNs may prove ineffective, as it can lead to inaccurate clustering results, heavy computational burden, and limited adaptability across diverse graph structures. Here we establish three fundamental designing principles of the clustering scheme for topology augmentation in GNNs.

- Enhancing the efficiency of clustering.
- Effectively incorporating label information from the current graph to guide clustering.
- Allowing for adaptive adjustment to diverse graph structures without relying on predefined assumptions.

Taking these principles into account, we propose a novel and efficient method, ClusterDrop, which aims to alleviate the issue of over-smoothing from a cluster-level perspective. It employs learnable prototypes to group nodes into multiple clusters, where a one-pass computation of the similarity between prototypes and node embeddings is achieved. The resulting clustering information is then leveraged to selectively remove edges. Moreover, we introduce a novel loss function in clustering to enhance the inherent smoothness of nodes belonging to the same class within each cluster. The learnability of prototypes sets our method apart from traditional clustering approaches, enabling the incorporation of supervised signals through additional loss terms. Consequently, our approach addresses both over-smoothing and adapts better to diverse graph structures. Empirical evaluation of six benchmark datasets demonstrates the superiority of ClusterDrop over existing techniques. Our contributions are summarized as follows.

1) We develop a scalable prototype-based clustering method to address the over-smoothing issue from a cluster-level view.

2) We propose a weighted random edge dropping strategy guided by clusters to prevent the smoothness of different class nodes.

3) We design a novel loss function to enhance the inherent smoothness of nodes within the same class in each cluster, leveraging supervised signals.

4) The proposed ClusterDrop is time-efficient and outperforms state-of-the-art methods for alleviating over-smoothing on six benchmark datasets.

## 2 Related Works

Existing methods for addressing over-smoothing fall into the following three categories, focusing on the model level, embedding level, and graph topology level, respectively.

**Mechanism modification.** JKNet [Xu *et al.*, 2018] proves the equivalence of graph convolution and random walks and identifies the over-smoothing problem as a missing initial information issue. JKNet adds jumping connections from shallow layers to the last layer to enhance the initial information. APPNP [Gasteiger *et al.*, 2019] also takes a view from random walks. It adopts personalized PageRank to preserve the initial node's local neighborhood and derives a message passing method based on the steady-state distribution of the personalized PageRank. Since the GCN layer is based on the first-order approximation of Chebyshev polynomials [Kipf and Welling, 2017], which equals a low pass filter on the spectral domain of graphs, multiple stacking of GCN layers filters out too many high-frequency input signals, leading to over-smoothing. Thus, GCNII [Chen *et al.*, 2020b] was proposed, which adds initial residual and identity mapping to approximate higher-order Chebyshev polynomials.

**Embedding regularization.** PairNorm [Zhao and Akoglu, 2020] is the first normalization method applied between intermediate layers to alleviate over-smoothing. The key idea is to keep pairwise distances unchanged, thus avoiding all node embeddings becoming too similar. ContraNorm [Guo *et al.*, 2023] recognizes the similarities between the over-smoothing issue in GNNs and the feature collapse issue in contrastive learning. ContraNorm proposes to alleviate over-smoothing from a contrastive view by tearing the InfoNCE loss into alignment loss and uniform loss. ContraNorm then transforms the process of optimizing the uniform loss with gradient descent into a model architecture to keep node embeddings distributed more uniformly.

**Topology augmentation.** This category of methods recognizes the crucial role of graph topology in the over-smoothing problem. DropEdge [Rong *et al.*, 2020] applies random edge dropping at training and theoretically proves that edge dropping helps alleviate the over-smoothing issue. AdaEdge [Chen *et al.*, 2020a] believes that the low information-to-noise ratio of nodes contributes to over-smoothing and proposes to identify intra/inter-class edges by iteratively training GNNs. In contrast to AdaEdge, GAUG [Zhao *et al.*, 2021] trains a graph auto-encoder as an edge prediction module and samples a new augmented adjacency matrix based on the module output. The training is then performed on the sampled adjacency matrix. GUIDE [Wang *et al.*, 2022] empirically finds that the frequency of each edge's occurrence as part of the shortest paths of all node pairs correlates with the edge types. GUIDE then proposes to precompute the occurrence frequency of each edge and use it as the weight to perform a weighted random edge dropping.

Our work also addresses over-smoothing through topology augmentation, but differs from previous methods by modeling the task from a cluster-level view, which better adapts to different graphs and is scalable to large graphs. Out of the scope of over-smoothing, our method shares a mathematical resemblance with [Hui *et al.*, 2021].

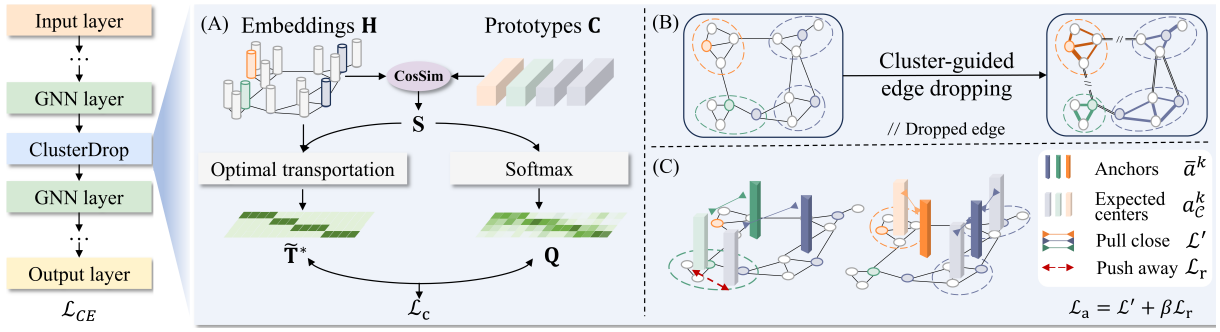



Figure 2: Overview of ClusterDrop. (A) Node clustering based on prototypes and optimal transportation. (B) Cluster-guided edge dropping. (C) Further enhancement of intra-class node smoothness.

## 3 Preliminary

This study focuses on the node classification problem. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be the input graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. Each node $i$ has an input features $\boldsymbol{x}_i \in \mathbb{R}^F$. Most of the current message passing GNN layers can be formulated as Eq. (1).

$$\boldsymbol{h}_i^{(l)} = f^{(l)}\left(\boldsymbol{h}_i^{(l-1)}, \text{aggr}_{j\in\mathcal{N}(i)}\left(\boldsymbol{h}_i^{(l-1)}, \boldsymbol{h}_j^{(l-1)}\right)\right), \quad (1)$$

where aggr is an aggregation function that aggregates neighbor messages, $\mathcal{N}(i)$ are the neighbors of node $i$, and $f^{(l)}$ is the function updating the $l$-layer node embedding $\boldsymbol{h}_i^{(l)}$. The initial node embedding $\boldsymbol{h}_i^{(0)}$ is equal to the input node feature $\boldsymbol{x}_i$. The final prediction label $\hat{y}_i$ is output by a decision function. Node classification tasks typically involve semi-supervised learning, where only a subset of nodes have labeled data available for training. The GNN layers are trained by the Cross-Entropy loss of $\hat{y}_i$ and the supervised labels.

## 4 The Framework of ClusterDrop

ClusterDrop functions as an intermediary module between GNN layers (Fig. 2). It consists of three major components. First, it takes node embeddings as input, computes their cosine similarity with prototypes, and derives a soft assignment matrix $\mathbf{Q}$. Optimal transportation is employed to enforce non-trivial $\mathbf{Q}$. Then, edge weights are derived from $\mathbf{Q}$, enabling weighted edge dropping. Additionally, we compute supervised class anchors and expected cluster centers, utilizing $\mathcal{L}_a$ to enhance the intra-class smoothness of nodes.

### 4.1 Node Clustering with Learnable Prototypes

Here we propose a prototype-based clustering method. For simplicity, we omit the superscript indicating the number of layers. Considering the $l$-th layers node embeddings $\mathbf{H} \in \mathbb{R}^{N\times F}$, where the $i$-th row is the embedding $\boldsymbol{h}_i$ of node $i$, $N = |\mathcal{V}|$ is the number of nodes, and $F$ is the dimensionality of embeddings. To cluster all nodes into $K$ clusters, we set $K$ prototypes $\mathbf{C} \in \mathbb{R}^{K\times F}$. We use cosine similarity to measure the preference of nodes to prototypes, then a soft assignment of nodes to the prototypes $\mathbf{Q} \in \mathbb{R}^{N\times K}$ can be computed by Eqs. (2-3).

$$\mathbf{S} = \text{cosine\_similarity}(\mathbf{H}, \mathbf{C}), \quad (2)$$

$$\mathbf{Q} = \text{softmax}(\mathbf{S}). \quad (3)$$

The similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times K}$ is computed between each node embedding and each prototype, and the softmax performs on the row vectors of $\mathbf{S}$. The hard assignment can be generated from the soft assignment by selecting the prototype with the highest similarity.

Since only similarity is considered in Eq. (3), one potential issue of the clustering is that the final assignment might be trivial, i.e., all nodes are assigned to the same prototype. The smooth nature of GNNs also exacerbates such collapse. To avoid this, we propose the uniform assignment of nodes to each prototype like [Caron *et al.*, 2020] did. This approach serves to restrict the similarity between node embeddings and alleviate the problem of over-smoothing. The assignment of nodes to prototypes under this constraint can be determined by solving an optimal transportation problem [Asano *et al.*, 2020; Villani, 2021].

Assuming that each node in the graph is of equal importance, the distribution of nodes can be represented by a uniform distribution. To avoid obtaining trivial solutions, the nodes should be uniformly distributed to each prototype, such that the resulting clusters can also be described by a uniform distribution. In this context, obtaining the constrained assignment is equivalent to identifying a transportation matrix $\mathbf{T} \in \mathbb{R}^{N\times K}$ that transports the uniform distribution of nodes to the uniform distribution of clusters with the cosine similarity $\mathbf{S}$ as the affinity matrix. Formally, the goal is to maximize the following optimal problem (Eq. (4)).

$$\begin{aligned} \max_{\mathbf{T}} \ & \text{Tr}(\mathbf{T}^\top \mathbf{S}) + \epsilon\mathcal{H}(\mathbf{T}) \\ \text{s.t. } & \mathbf{T}\mathbf{1} = N^{-1}\mathbf{1}, \\ & \mathbf{T}^\top\mathbf{1} = K^{-1}\mathbf{1}, \end{aligned} \quad (4)$$

where $\mathbf{1}$ is an all one vector, $\epsilon$ is a constant, and $\mathcal{H}(\mathbf{T})$ computes the entropy of $\mathbf{T}$. We add this entropy term, thus the solution $\mathbf{T}^*$ of this problem can be found by Sinkhorn-Knopp algorithm [Sinkhorn and Knopp, 1967] efficiently.

To avoid the collapse and relieve the over-smoothing, the actual assignment $\mathbf{Q}$ should be close to the constrained assignment. Thus, the prototypes are trained by the loss below,

$$\mathcal{L}_c = \left\langle \tilde{\mathbf{T}}^*, \log \mathbf{Q} \right\rangle, \quad (5)$$

where $\tilde{\mathbf{T}}^*$ is the one-hot version of $\mathbf{T}^*$ by setting the max row value to 1, and $\langle\cdot,\cdot\rangle$ means dot product of matrices.

### 4.2 Cluster-Guided Edge Dropping

To avoid the smoothness between groups of different classes (the first aspect mentioned in Section 1), we prioritize the removal of edges linking nodes with different labels. The clusters obtained through the use of prototypes can guide this process by indicating which edges should be removed first. Specifically, if two nodes are more likely to be assigned to different clusters, they are more likely to belong to different classes. The dissimilarity can be measured by a cluster-level distance as described below. With the soft assignment $\mathbf{Q}$, each node can be coded by the row of $\mathbf{Q}$ as a distribution to the prototypes. Then the cluster-level distance of two nodes can be measured by Janson-Shannon divergence as shown in Eq. (6).

$$\text{dist}(i,j) = \frac{1}{2}\left(\text{KL}(\boldsymbol{q}_i||\boldsymbol{q}_j) + \text{KL}(\boldsymbol{q}_j||\boldsymbol{q}_i)\right), \tag{6}$$

where $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ are the memebership distribution of node $i$ and node $j$ respectively, i.e., the $i$-th row and $j$-th row of $\mathbf{Q}$. Then each edge $(i,j)$ is associated with a weight $w_{ij}$,

$$w_{ij} = \frac{\text{dist}(i,j)}{\sum_{(k,l)\in\mathcal{E}} \text{dist}(k,l)}. \tag{7}$$

To relieve the over-smoothing issue, a weighted random sampling based on $w_{ij}$ is performed to drop some edges from the original graph in the training phase. Notably, the self-loop is naturally excluded from the random drop because $w_{ij}$ is always equal to zero when $i = j$.

### 4.3 Smoothing Embeddings with Supervised Signals

In addition to preventing the blurring of clusters, we enhance the smoothness of nodes within the same class to achieve better performance (the second aspect mentioned in Section 1). In the semi-supervised setting, the nodes used for training are provided with supervised information. The average embedding of these supervised nodes can serve as anchors for the corresponding classes. Through clustering, these nodes are assigned to several clusters and act as bridges between the cluster space and the target label space. To enhance smoothness within the same class, we can pull the embeddings from a given cluster closer to the most similar supervised anchor.

Let $\mathcal{V}_{\text{tr}} \subset \mathcal{V}$ be the node set with known labels. The anchor of class $k$ can be computed by Eq. (8).

$$\bar{\boldsymbol{a}}^k = \frac{1}{|\{i|i \in \mathcal{V}_{\text{tr}}, y_i = k\}|} \sum_{i\in\mathcal{V}_{\text{tr}}, y_i=k} \boldsymbol{h}_i, \tag{8}$$

where $y_i$ is the label of node $i$.

Then we compute the expected centers of a cluster. Considering a cluster $\mathcal{C} \subset \mathcal{V}$ with some nodes in $\mathcal{V}_{\text{tr}}$, i.e., $\mathcal{C} \cap \mathcal{V}_{\text{tr}} \neq \emptyset$. The labeled nodes in $\mathcal{C}$ are the category references for the other nodes in $\mathcal{C}$. We assume that the probability of a node being assigned to a class $k$ follows a Gaussian distribution based on the distance between the node and the center of the label nodes (of class $k$) within that cluster. Thus, the expected center of class $k$, $\boldsymbol{a}_{\mathcal{C}}^k$, can be computed by the following Eqs. (9-11), where $\tau$ is a coefficient representing the variance.

$$\boldsymbol{a}_{\mathcal{C}}^k = \sum_{i\in\mathcal{C}, i\notin\mathcal{V}_{\text{tr}}} w_i^k \boldsymbol{h}_i, \tag{9}$$

$$w_i^k = \frac{\exp(-\|\boldsymbol{h}_i - \boldsymbol{h}_{\text{tr}}^k\|_2^2/\tau)}{\sum_{j\in\mathcal{C}, j\notin\mathcal{V}_{\text{tr}}} \exp(-\|\boldsymbol{h}_j - \boldsymbol{h}_{\text{tr}}^k\|_2^2/\tau)}, \tag{10}$$

$$\boldsymbol{h}_{\text{tr}}^k = \frac{1}{|\{i|i \in \mathcal{C} \cap \mathcal{V}_{\text{tr}}, y_i = k\}|} \sum_{i\in\mathcal{C}\cap\mathcal{V}_{\text{tr}}, y_i=k} \boldsymbol{h}_i. \tag{11}$$

To enhance the smoothness of node embeddings within the same class, we pull the expected centers of each cluster closer to the corresponding class anchors. Given that nodes within a cluster are more likely to belong to the class which has the most supervised nodes within that cluster, the loss function should reflect this preference. The loss function is formulated as a weighted sum, as shown in Eqs. (12-13), where $s_{\mathcal{C}}^k$ denotes the weight of class $k$ for cluster $\mathcal{C}$.

$$\mathcal{L}' = \sum_{\{\mathcal{C}|\mathcal{C}\cap\mathcal{V}_{\text{tr}}\neq\emptyset\}} \sum_k s_{\mathcal{C}}^k \|\boldsymbol{a}_{\mathcal{C}}^k - \bar{\boldsymbol{a}}^k\|_2^2, \tag{12}$$

$$s_{\mathcal{C}}^k = |\{i|i \in \mathcal{C} \cap \mathcal{V}_{\text{tr}}|y_i = k\}| \,/\, |\{i|i \in \mathcal{C} \cap \mathcal{V}_{\text{tr}}\}| \tag{13}$$

As we aim to form clusters that are as pure as possible, with each cluster containing the majority of nodes from a single class. Thus, a regularized term $\mathcal{L}_{\text{r}}$ is added to push away the expected centers of different classes, i.e.,

$$\mathcal{L}_{\text{r}} = \sum_{\{\mathcal{C}|\mathcal{C}\cap\mathcal{V}_{\text{tr}}\neq\emptyset\}} \sum_{k_i\neq k_j} {\boldsymbol{a}_{\mathcal{C}}^{k_i}}^{\top} \boldsymbol{a}_{\mathcal{C}}^{k_j}. \tag{14}$$

Then the total loss is expressed in Eq. (15). We call it assignment loss because the smoothness of nodes from the same class is achieved by assigning the expected centers of clusters to the corresponding supervised anchors.

$$\mathcal{L}_{\text{a}} = \mathcal{L}' + \beta\mathcal{L}_{\text{r}}. \tag{15}$$

With the GNN classification loss $\mathcal{L}_{\text{CE}}$, the overall loss to trained ClusterDrop and GNNs simultaneously is formulated in Eq. (16). The $\beta$, $\alpha_1$ and $\alpha_2$ are coefficients.

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \alpha_1\mathcal{L}_{\text{c}} + \alpha_2\mathcal{L}_{\text{a}} \tag{16}$$

## 5 Theoretical Support of ClusterDrop

Previous works of topology augmentation empirically find out that removing inter-class edges is crucial in alleviating the over-smoothing problem [Chen *et al.*, 2020a; Zhao *et al.*, 2021; Wang *et al.*, 2022]. Here we show that dropping the inter-class edges theoretically helps relieve the over-smoothing. Specifically, we prove that after passing through a sufficient number of GCN layers, nodes remain distinguishable even after all inter-class edges are removed.

**Theorem 1.** *Given a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $K$-class nodes, assume that for every pair of nodes within a class $k$, there exists a path connecting the pair with nodes only from class $k$. Then, after applying a sufficient number of GCN layers, the input node embedding $\mathbf{X}^{(0)} \in \mathbb{R}^{N\times F}$ will converge to $K$ directions represented by vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K$ with respect to their respective classes, in the circumstance that all inter-class edges are removed from the original graph.*

| Backbone | Method | Cora | Citeseer | Pubmed | Chameleon | USA-Airport |
|---|---|---|---|---|---|---|
| GCN | $k$-means | 83.7±0.5 | 72.2±0.4 | 79.6±0.4 | 60.6±0.4 | 60.9±0.7 |
| | Vanilla | 81.2±0.5 | 70.8±0.8 | 78.9±0.4 | 58.7±1.5 | 56.1±0.8 |
| | DropEdge | 82.3±0.3 | 71.7±0.5 | 78.9±0.4 | 58.7±0.8 | 56.6±0.8 |
| | AdaEdge | 81.9±0.7 | 72.8±0.7 | 79.6±0.3 | 59.7±1.3 | 57.7±0.8 |
| | GUIDE | 82.1±0.6 | 72.2±0.3 | OOT | 58.8±1.0 | 57.5±0.6 |
| | GAUG | 83.6±0.5 | **73.3±1.0** | 77.3±0.5 | 59.9±0.6 | 61.4±0.9 |
| | ClusterDrop | **84.3±0.6** | 72.7±0.6 | **79.8±0.4** | **65.9±0.6** | **62.3±0.6** |
| GraphSAGE | Vanilla | 81.1±0.8 | 70.5±1.5 | 78.2±0.8 | 62.9±1.2 | 59.5±1.1 |
| | DropEdge | 80.2±0.5 | 70.4±1.0 | 78.4±0.9 | 60.2±0.5 | 59.3±0.9 |
| | AdaEdge | 81.5±0.6 | 71.3±0.8 | 78.4±0.2 | 59.9±1.0 | 57.4±0.5 |
| | GUIDE | 81.2±1.0 | 70.5±0.6 | OOT | 61.9±1.6 | 58.2±0.8 |
| | GAUG | 82.0±0.5 | 72.7±0.7 | 78.8±0.5 | 60.7±0.6 | 57.1±0.7 |
| | ClusterDrop | **82.9±0.6** | **72.7±0.6** | **79.7±1.0** | **66.7±1.3** | **60.6±0.6** |
| JKNet | Vanilla | 77.0±1.6 | 69.3±0.4 | 78.3±0.7 | 63.7±1.4 | 59.5±0.6 |
| | DropEdge | 79.8±1.4 | 69.5±1.0 | 78.4±0.5 | 65.0±1.0 | 60.0±1.0 |
| | Adadge | 80.4±1.4 | 68.9±1.2 | 78.5±1.1 | 64.9±0.9 | 59.5±0.5 |
| | GUIDE | 80.8±1.4 | 69.3±0.8 | OOT | 62.0±1.2 | 59.4±1.1 |
| | GAUG | 80.5±0.9 | 69.7±1.4 | 77.6±0.6 | 64.9±1.0 | 60.4±1.0 |
| | ClusterDrop | **82.0±0.5** | **71.3±0.6** | **79.5±0.7** | **67.0±0.9** | **62.3±0.8** |
| GAT | Vanilla | 81.0±0.7 | 70.4±0.7 | 77.8±0.5 | 55.2±2.1 | 54.7±1.4 |
| | DropEdge | 82.0±0.9 | 71.3±0.6 | 78.2±0.4 | 54.7±1.5 | 54.8±1.4 |
| | AdaEdge | 82.0±0.6 | 71.1±0.8 | 78.6±0.5 | 58.2±1.8 | 56.7±1.0 |
| | GUIDE | 82.3±0.6 | 71.3±1.2 | OOT | 56.3±1.7 | 53.3±0.7 |
| | GAUG | 82.2±0.8 | 71.6±1.1 | OOM | 64.9±1.1 | 54.6±1.1 |
| | ClusterDrop | **82.7±0.7** | **72.6±0.6** | **79.5±0.5** | **67.2±1.7** | **57.0±1.6** |

Table 1: Comparison results with different GNN backbones. The compared SOTA models include DropEdge (ICLR-20'), AdaEdge (AAAI-20'), GAUG (AAAI-21'), and GUIDE (TNNLS-22').

The detailed proof of Theorem 1 is provided in Appendix. From Theorem 1, it is easy to see that the nodes can be classified into $K$ classes without performance degradation. This shows that dropping the inter-class edges can cut off the mixture of messages from different classes, thereby alleviating the over-smoothing issue. The cluster-guided edge dropping technique employed in ClusterDrop incorporates this finding. Specifically, since nodes with higher similarity are clustered into the same group, edges dropped between clusters are more likely to result in the removal of inter-class edges. Our experiments demonstrate the effectiveness of this approach, and the results are consistent with [Wu *et al.*, 2023b]. Moreover, the number of prototypes in cluster-guided edge dropping also serves as regularization to restrict the lower bound of $K$, thus making features more diverse to relieve the over-smoothing.

## 6 Time Complexity Analysis

In this section, we provide a time complexity analysis of the proposed ClusterDrop. ClusterDrop is a training-time augmentation module thus bringing no cost at inference time. During training, the additional cost of ClusterDrop arises from three parts, computing $\mathcal{L}_{\mathrm{c}}$, computing $\mathcal{L}_{\mathrm{a}}$, and performing weighted random drop. Computing $\mathcal{L}_{\mathrm{c}}$ includes the computation of node assignment $\mathbf{Q}$ and solving the optimal transportation problem, where the former takes $\mathrm{O}(K|\mathcal{V}|)$ and the latter takes $\mathrm{O}(mK|\mathcal{V}|)$, where $m$ is the number of iterations to perform the Sinkhorn-Knopp algorithm and empirically it is smaller than 10. As Eqs. (8-15) show, the computation of $\mathcal{L}_{\mathrm{a}}$ is linear with the number of nodes with time complexity $\mathrm{O}(|\mathcal{V}|)$. The weighted random drop is performed on existing edges, and the sampling algorithm has an implementation with $\mathrm{O}(p|\mathcal{E}|\log|\mathcal{E}|)$ [Wong and Easton, 1980], where $p$ is the dropping rate. Therefore, in summary, ClusterDrop brings additional $\mathrm{O}(mK|\mathcal{V}|+p|\mathcal{E}|\log|\mathcal{E}|)$ time complexity in training time, which brings little burden compared to common GNNs with $\mathrm{O}(|\mathcal{V}|+|\mathcal{E}|)$ time complexity, and no additional cost in the inference time.

## 7 Experiments

We evaluate the performance of ClusterDrop on 6 benchmark datasets covering 3 real-world scenarios [Kipf and Welling, 2017; Pei *et al.*, 2020; Wu *et al.*, 2019; Hu *et al.*, 2020] and compare with 4 state-of-the-art methods, namely DropEdge [Rong *et al.*, 2020], AdaEdge [Chen *et al.*, 2020a], GUIDE [Wang *et al.*, 2022], and GAUG [Zhao *et al.*, 2021], on 4 widely used GNN backbones. More details of datasets and implementation are provided in Appendix. Codes are available on https://github.com/YYHemich/ClusterDrop.

### 7.1 ClusterDrop Improves GNNs' Performance

The results of the accuracy comparison are shown in Table 1. The results of GUIDE on Pubmed are not available due to the prohibitively high time cost of precomputing required for this dataset. This is denoted as OOT (Out-Of-Time). A detailed analysis of the time complexity of GUIDE is provided in Appendix. The missing result of GAUG with GAT on Pubmed is

due to out-of-memory (OOM). As shown in Table 1, ClusterDrop significantly enhances vanilla GNNs across five benchmark datasets and outperforms recent topological augmentation methods with most of GNN backbones, suggesting the robustness of ClusterDrop in different application scenarios.

**Comparison to uniformly random dropping.** DropEdge is a wide-used edge-dropping method for GNNs, which randomly drops edges from graphs with a certain probability. ClusterDrop outperforms DropEdge with all 4 GNN backbones. This indicates that the weighted dropping strategy guided by prototype clustering prunes noise edges more accurately. While DropEdge generally performs better than the original GNNs, we notice a decrease in performance in some cases, such as the results of GraphSAGE on Citeseer and Chameleon datasets. This may be due to several bad random drops during training without guidance. By contrast, ClusterDrop employs a guided dropping approach to enhance topology augmentations, resulting in more stable improvements.

**Comparison to unsupervised guidance.** Here we compare ClusterDrop with GUIDE [Wang *et al.*, 2022], an unsupervised assumption-based approach that employs a weighted edge dropping strategy. GUIDE assumes the intra/inter-class edges are correlated with the numbers of their occurrence in all pairwise shortest paths of a graph, and it drop edges weighted by these numbers. As shown in Table 1, ClusterDrop achieves higher accuracy than GUIDE in all comparisons. Particularly, ClusterDrop exhibits an average improvement of 6.9% on Chameleon, indicating that the assumption of GUIDE may not suit all cases. We also compare the prototype clustering with the traditional clustering method. We replace the prototype clustering of ClusterDrop by $k$-means (results in Table 1) and observe that it outperforms DropEdge and GUIDE while performing on par with edge predictor methods GAUG and AdaEdge. This finding further validates the effectiveness of cluster guidance. However, $k$-means clustering does not perform as well as ClusterDrop because ClusterDrop is more flexible in adapting to graphs by learning from them directly. Besides, ClusterDrop is more computationally efficient, since $k$-means takes a lot of iterations to converge, particularly on large graph datasets (with more than 100 iterations).

**Comparision to learning edge predictor.** Another popular series is graph augmentation through learning an edge predictor, which models the distribution of the adjacent matrix and samples new graph topology from the distribution to train GNNs. We first compare ClusterDrop with GAUG. ClusterDrop performs better on 4 of the 5 datasets. While ClusterDrop does not always yield the optimal performance across different backbones on the Citeseer dataset, it still manages to achieve the highest result on JKNet and GAT, which possess a greater number of parameters compared to GCN and GraphSAGE. This indicates that, besides relieving over-smoothing, the cluster-guided edge dropping of ClusterDrop also provides stronger regularization to avoid over-fitting. Furthermore, ClusterDrop is more memory-efficient than GAUG, enhancing its suitability for large graph training, while GAUG faces out-of-memory (OOM) issues when using GAT as the backbone on Pubmed in our experiments. AdaEdge can be seen as a special case to train edge predictors, which iteratively trains GNNs to remove/add edges. ClusterDrop outperforms AdaEdge and exhibits greater training efficiency, as it does not require iterative training.

## 7.2 Relieving Over-smoothing for Deep GNNs

Over-smoothing often leads to performance degradation when stacking multiple GNN layers. In this experiment, we demonstrate how ClusterDrop can alleviate this issue. We compare the accuracy of GCNs with 2, 4, 6, and 8 layers, each equipped with different methods. Here ClusterDrop is applied once at the first GCN layer.

The experiment is performed on the Cora dataset and the results are shown in Fig. 3(a). We observe that as the GNN

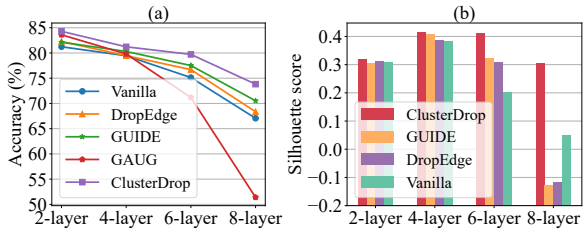



Figure 3: Performance variation across different depths of GCNs on Cora. The silhouette score is computed based on cosine distance.

becomes deeper, the overall accuracy decreases due to the gradually severe over-smoothing issue. As shown in Fig. 3(a), ClusterDrop maintains a substantial advantage in different depths of the GCN compared to other methods, indicating its superior performance in relieving over-smoothing. Furthermore, the degradation of ClusterDrop is slower, illustrating its capability to delay over-smoothing.

As we mentioned earlier, relieving over-smoothing involves two aspects: avoiding smoothness between the node groups of different classes and preserving smoothness within groups of nodes of the same class. Here we demonstrate how ClusterDrop addresses both of these aspects by computing the silhouette score [Rousseeuw, 1987], which measures the tightness of each cluster and the difference between clusters. We compute the silhouette score of the last hidden embeddings, the higher silhouette score indicates a tighter cluster and a greater difference between clusters. As shown in Fig. 3(b), ClusterDrop achieves a higher silhouette score than other methods, and the performance gap becomes more obvious with increasing depth. This demonstrates that ClusterDrop learns better embeddings that represent label clusters and its clustering scheme is effective in alleviating over-smoothing in deep GNNs.

## 7.3 Efficacy in Dropping Inter-Class Edges

As Theorem 1 and previous studies [Chen *et al.*, 2020a; Zhao *et al.*, 2021; Wang *et al.*, 2022] asserted, the removal of inter-class edges plays a crucial role in influencing the prediction performance of GNNs and the extent of over-smoothing that may occur. Here we provide insight into the effectiveness of ClusterDrop by examining the ratio of inter-class edges among the dropped edges during GCN training on Cora and compare the average number of dropped inter-class edges across different methods at varying drop rates.

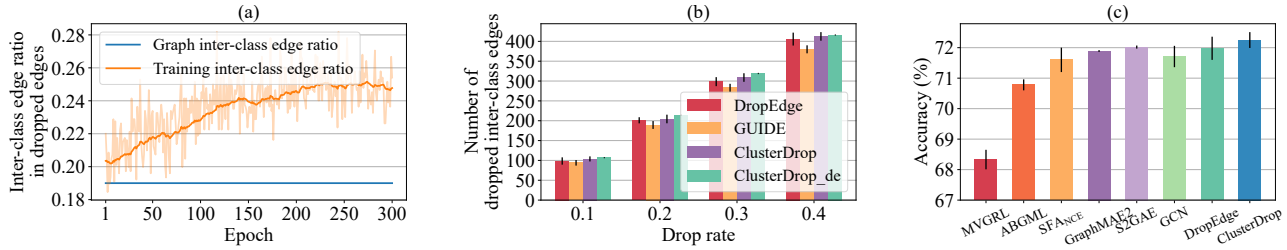



Figure 4: (a) and (b) are ratios and numbers of dropped inter-class edges under different training epochs and drop rate settings. Due to the randomness in dropping, the inter-class edge dropping ratio fluctuates (the light orange curve in (a)) . The deep orange curve shows exponentially smoothed values. (c) Results on ogbn-arxiv dataset. The compared GSSL models include MVGRL (ICML-20'), $\text{SFA}_{\text{NCE}}$ (AAAI-23'), S2GAE (WSDM-23'), ABGML, and GraphMAE2 (both WWW-23').

Fig. 4(a) shows the ratio of dropped inter-class edges, which gradually increases as the training epochs progress and eventually converges around 25%. The blue curve represents the expected ratio of inter-class edges in a uniform sampling. This suggests that the cluster-level distance effectively weights inter-class edges with a high dropping probability.

We also compare the number of dropped inter-class edges of ClusterDrop with DropEdge, GUIDE, and a variant of ClusterDrop named ClusterDrop_de, in which the weighted random drop is replaced with a deterministic approach that removes edges in a descending order of their weights, from highest to lowest. As shown in Fig. 4(b), ClusterDrop drops more inter-class edges than DropEdge and GUIDE, indicating that the clusters found by prototypes indeed capture label correlated information and provide better guidance than random drop. ClusterDrop_de removes the most inter-class edges, further suggesting that ClusterDrop assigns more inter-class edges with higher weights than a totally random approach thus weighting edges better according to edge types. However, ClusterDrop with a weighted random drop is still preferable in training because the deterministic drop without randomness is more likely to overfit and largely depends on the initialization. A poor initialization may lead to a false topology augmentation as investigated in [Chen *et al.*, 2020a].

### 7.4 Performance on Large Graphs

We test ClusterDrop on a large-scale dataset, Ogbn-arxiv, to show it is also scalable on larger graphs, in which many edge-predictor-based methods are difficult to apply with significant memory costs. Due to the non-scalability of other methods, only DropEdge is compared. Instead, we compare several recent graph self-supervised learning (GSSL) methods, which achieve outstanding performance on small graphs [Hassani and Khasahmadi, 2020; Chen *et al.*, 2023; Zhang *et al.*, 2023; Hou *et al.*, 2023; Tan *et al.*, 2023]. As shown in Fig. 4(c), generally GSSL methods do not perform superiorly on the large-scale dataset, and the proposed ClusterDrop performs the best compared with the best GSSL method S2GAE and topology augmentation method DropEdge.

### 7.5 Ablation Study of the Auxiliary Losses

ClusterDrop is trained with two auxiliary losses $\mathcal{L}_{\text{c}}$ and $\mathcal{L}_{\text{a}}$, where $\mathcal{L}_{\text{a}}$ includes a regularized term $\mathcal{L}_{\text{r}}$ (Eq. (14)). We perform ablation studies with 2-layer GCNs on the citation datasets to explore the effects of different losses. Table 2 shows the ablation results. Since the prototypes are mainly trained by $\mathcal{L}_{\text{c}}$, ClusterDrop without $\mathcal{L}_{\text{c}}$ degenerates to random drop like DropEdge. ClusterDrop with $\mathcal{L}_{\text{c}}$ enhances the performance of the vanilla GCN, showing the effectiveness of cluster-guided edge dropping in mitigating over-smoothing. Moreover, it also outperforms the model without $\mathcal{L}_{\text{c}}$, demonstrating the superiority of cluster-guided edge dropping over totally random drops again. When comparing ClusterDrop with and without $\mathcal{L}_{\text{a}}$, improvements in accuracy are observed across all three datasets, highlighting the advantage of the assignment loss in ClusterDrop. These findings emphasize the importance of preserving the smoothness of nodes in the same classes in addressing the over-smoothing issue. Moreover, the comparison between ClusterDrop with and without $\mathcal{L}_{\text{r}}$ demonstrates that the regularized term further improves the accuracy, indicating that the design of $\mathcal{L}_{\text{r}}$ to enhance the purity of clusters by pushing away the embeddings of different classes is effective.

| | $\mathcal{L}_{\text{c}}$ | $\mathcal{L}_{\text{a}}$ | $\mathcal{L}_{\text{r}}$ | Cora | Citeseer | Pubmed |
|---|---|---|---|---|---|---|
| GCN | - | - | - | 81.2±0.6 | 70.8±0.8 | 78.9±0.4 |
| ours | ✗ | - | - | 82.3±0.6 | 71.5±0.4 | 78.9±0.4 |
| | ✓ | ✗ | - | 82.3±0.3 | 72.3±0.4 | 78.9±0.4 |
| | ✓ | ✓ | ✗ | 83.3±0.4 | 72.6±0.4 | 79.3±0.4 |
| | ✓ | ✓ | ✓ | **84.3±0.6** | **72.7±0.6** | **79.8±0.4** |

Table 2: Ablation study results of auxiliary losses.

## 8 Conclusion

GNN models commonly suffer from the over-smoothing issue. While a lot of previous works have attempted to alleviate this issue through topology augmentation, determining a suitable augmentation method remains challenging. To address this challenge, we propose a novel method called ClusterDrop, which aims to alleviate the over-smoothing issue from an efficient and effective cluster perspective. Specifically, our method clusters nodes based on learnable prototypes and then performs a cluster-level weighted edge dropping during training. The assignment loss is introduced to enhance the smoothness of nodes with the same label. Extensive experiments on six benchmark datasets demonstrate that it outperforms SOTA methods and is particularly effective in relieving the over-smoothing issue.

## Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2023YFC2811502) and the National Natural Science Foundation of China (Nos. 62272300 and 61972251).

## References

[Asano *et al.*, 2020] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Caron *et al.*, 2020] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Chen *et al.*, 2020a] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[Chen *et al.*, 2020b] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[Chen *et al.*, 2023] Dong Chen, Xiang Zhao, Wei Wang, Zhen Tan, and Weidong Xiao. Graph self-supervised learning with augmentation-aware contrastive learning. In *Proceedings of the ACM Web Conference 2023*, pages 154–164, 2023.

[Ding *et al.*, 2019] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703. Association for Computational Linguistics, July 2019.

[Gasteiger *et al.*, 2019] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019.

[Guo *et al.*, 2023] Xiaojun Guo, Yifei Wang, Tianqi Du, and Yisen Wang. Contranorm: A contrastive learning perspective on oversmoothing and beyond. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

[Hassani and Khasahmadi, 2020] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

[Hou *et al.*, 2023] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM Web Conference 2023*, pages 737–746, 2023.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Hui *et al.*, 2021] Bo Hui, Da Yan, and Wei-Shinn Ku. Node-polysemy aware recommendation by matrix completion with side information. In *2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021*, pages 636–642. IEEE, 2021.

[Jiang *et al.*, 2022] Xuan Jiang, Zhiyong Yang, Peisong Wen, Li Su, and Qingming Huang. A sparse-motif ensemble graph convolutional network against over-smoothing. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2094–2100. ijcai.org, 2022.

[Keriven, 2022] Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over)smoothing. In *NeurIPS*, 2022.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Kipf *et al.*, 2018] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Liu *et al.*, 2023] Yang Liu, Chuan Zhou, Shirui Pan, Jia Wu, Zhao Li, Hongyang Chen, and Peng Zhang. Curvdrop: A ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 221–230, 2023.

[Newman, 2006] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[Pei *et al.*, 2020] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International*

*Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Rong *et al.*, 2020] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[Rousseeuw, 1987] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[Sinkhorn and Knopp, 1967] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[Tan *et al.*, 2023] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 787–795, 2023.

[Villani, 2021] Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

[Wang *et al.*, 2022] Jie Wang, Jianqing Liang, Jiye Liang, and Kaixuan Yao. Guide: Training deep graph neural networks via guided dropout over edges. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[Wong and Easton, 1980] Chak-Kuen Wong and Malcolm C. Easton. An efficient method for weighted sampling without replacement. *SIAM Journal on Computing*, 9(1):111–113, 1980.

[Wu *et al.*, 2019] Jun Wu, Jingrui He, and Jiejun Xu. Demonet: Degree-specific graph neural networks for node and graph classification. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 406–415. ACM, 2019.

[Wu *et al.*, 2023a] Gongce Wu, Shukuan Lin, Xiaoxue Shao, Peng Zhang, and Jianzhong Qiao. QPGCN: graph convolutional network with a quadratic polynomial filter for overcoming over-smoothing. *Appl. Intell.*, 53(6):7216–7231, 2023.

[Wu *et al.*, 2023b] Xinyi Wu, Zhengdao Chen, William Wei Wang, and Ali Jadbabaie. A non-asymptotic analysis of oversmoothing in graph neural networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Xu *et al.*, 2018] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

[Zachary, 1977] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

[Zhang *et al.*, 2023] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral feature augmentation for graph contrastive learning and beyond. pages 11289–11297. AAAI Press, 2023.

[Zhao and Akoglu, 2020] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

[Zhao *et al.*, 2021] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023, 2021.

[Zheng *et al.*, 2020] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. GMAN: A graph multi-attention network for traffic prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1234–1241. AAAI Press, 2020.