Structure-Aware DropEdge Towards Deep Graph Convolutional Networks

Jiaqi Han, Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, Junzhou Huang

Abstract—**It has been discovered that Graph Convolutional Networks (GCNs) encounter a remarkable drop in performance when multiple layers are piled up. The main factor that accounts for why deep GCNs fail lies in *over-smoothing*, which isolates the network output from the input with the increase of network depth, weakening expressivity and trainability. In this paper, we start by investigating refined measures upon DropEdge—an existing simple yet effective technique to relieve over-smoothing. We term our method as DropEdge++ for its two structure-aware samplers in contrast to DropEdge: *layer-dependent sampler* and *feature-dependent sampler*. Regarding the layer-dependent sampler, we interestingly find that increasingly sampling edges from the bottom layer yields superior performance than the decreasing counterpart as well as DropEdge. We theoretically reveal this phenomenon with Mean-Edge-Number (MEN), a metric closely related to over-smoothing. For the feature-dependent sampler, we associate the edge sampling probability with the feature similarity of node pairs, and prove that it further correlates the convergence subspace of the output layer with the input features. Extensive experiments on several node classification benchmarks, including both full- and semi- supervised tasks, illustrate the efficacy of DropEdge++ and its compatibility with a variety of backbones by achieving generally better performance over DropEdge and the no-drop version.**

Index Terms—**Graph Convolutional Network, DropEdge++, Layer-Dependent sampler, Feature-Dependent sampler, Mean-Edge-Number.**

I. INTRODUCTION

GRAPH Convolutional Networks (GCNs) have recently dominated the analysis of graph structures on a variety of tasks, *e.g.* node classification [1]. Against those based on graph kernels [2] or graph embedding [3], [4], the crucial benefit of using GCNs stems from that it is capable of representing both graph topology and node features in an end-to-end training manner. Whilst much progress has been made since [1], stacking more layers could be the most natural way to deliver a more expressive model that is expected to capture wider neighborhood proximity of each node. However, the performance of deep GCNs (even equipped with residual connections [5]) is much worse than their shallow variants on node classification [1].

Jiaqi Han, and Fuchun Sun are with the Department of Computer Science and Technology, Tsinghua Univerisity, Beijing, China.

Wenbing Huang is with Institute for AI Industry Research (AIR), Tsinghua University.

Yu Rong, Tingyang Xu, and Junzhou Huang are with Tencent AI Lab, Shenzhen, China.

The most acknowledged answer to why deep GCNs fail is *over-smoothing*. As firstly proposed by [6], over-smoothing drives the output of GCNs towards a stationary point (more precisely, a low-dimension subspace as proved by [7]) that contains limited topology information (*i.e.* node degree) regardless of input features with the growing number of layers stacked. This property, unavoidably, causes gradient vanishing and hinders the training of deep GCNs. There have been developed various methods to alleviate such issue, including the architecture-based methods [8], [9], [10] and the operation-based methods [11], [12], [13], [14]. We specify the introduction of these methods in § II.

Among the techniques proposed so far, DropEdge [12] uniformly drops out a certain number of edges at each training iteration, which is demonstrated to relieve over-smoothing theoretically and practically, and is able to enhance the performance of various kinds of deep GCNs. Yet and still, the improvement of deep GCNs by DropEdge over shallow models is marginal, leaving room to more powerful development. The main bottleneck of DropEdge is that its existing sampling strategy is unable to take full advantage of the structural information among the network design and data distribution: 1) the sampling of the adjacency matrix is shared across each layer (or sampled i.i.d. for the layer-wise version [12]), hence the relation of sampling behavior between consecutive layers is never characterized; 2) it will neglect the different roles of different edges by sampling each edge uniformly.

In this paper, we put a step further and propose *DropEdge++*, an enhanced version of DropEdge. DropEdge++ is composed of two structure-aware enhancements: *layer-dependent sampler* (inter-layer) and *feature-dependent sampler* (intra-layer). The layer-dependent sampler assumes the inductive bias that the adjacency matrix of the current layer is created by dropping edges conditional on the upper layer: *the Layer-Increasingly-Dependent (LID) style*, or conditional on the lower layer: *the Layer-Decreasingly-Dependent (LDD) style*. Interestingly, our experiments reveal that LID boosts the training of deep GCNs much better than LDD and DropEdge. This result implies that message passing over nodes in bottom layers are more vulnerable to over-smoothing and the edges in-between should be removed more than those of the top layers. We define a novel concept of Mean-Edge-Number (MEN) and show that LID derives smaller MEN than all others. The detailed explanations are provided in § IV-A.

Our second contribution—Feature-Dependent (FD) sampler leverages a certain kernel to measure the feature similarity of each adjacent node pair, and the normalized kernel value over all edges is then assigned as the edge sampling rate. As we

will draw in this paper, our FD not only inherits all advantages of DropEdge on alleviating over-smoothing but also correlates the output of deep GCNs with the input features even when the number of layers approaches the infinity. More details are presented in § IV-B.

DropEdge++ combining both LID and FD is compatible with other techniques for better performance including the architecture-based refinements [8], [9], [10]. Extensive evaluations on 6 popular benchmarks of node classification demonstrate that DropEdge++ remarkably outperforms DropEdge and the vanilla models without any sampling on varying backbones of both deep and shallow architectures. Our experiments also find that the best performance is usually observed on deep models with the depth beyond 2, if leveraging DropEdge++ to a carefully selected backbone.

II. RELATED WORK

Graph Neural Networks There have been diverse attempts with graph neural networks in literature. The first prominent research [15] builds the seminal graph convolutional model in both the spatial and spectral views. Following [15], many works [1], [16], [17], [18], [19] have been done for performance enhancement and/or efficiency obtainment. Among them, the work by [1] gives an efficient approximation to conduct the spectral operation in the spatial domain by aggregating the information from neighbor nodes. Later, GAT [20] introduces the attention mechanism to extend the learnability of message aggregation. KerGNN [21] integrates graph kernels into the message passing process of GNNs, by adopting trainable hidden graphs as graph filters which are combined with subgraphs to update node embeddings using graph kernels. Although KerGNN also aims at enhancing the expressivity of GNNs similar to our DropEdge++, the fundamental goal of our paper is different and it is to investigate how to address the over-smoothing issue caused by the increase of depth. Notably, we also apply kernels for edge sampling in the feature-dependent sampler, but the kernels here are used in a different way compared to KerGNN. Recently, to reduce the computational overhead of GCNs, various node-sampling-based methods are developed [22], [23], [24], [25], including the node-wise sampling [22], layer-wise sampling [26], [27], and subgraph-wise sampling [28].

Relieving Over-Smoothing It has been revealed that GCNs encounter a significant drop in performance with the depth beyond two [1]. As first introduced in [6] and recently generalized by [7], over-smoothing stands in the way of building deep GCNs that the output converges to a subspace which is irrelevant to the input features. On aware of this, continuous efforts have been made among which the architecture-based and operation-based methods are two popular genres. The architecture-based methods refine deep GCNs by adding dense skip connections from intermediate layers to the output (*i.e.* JKNet by [8]), or reversely from the input layer to intermediate layers (*i.e.* APPNP by [9]). By this means, the activations in lower layers are preserved, thus capable of relieving over-smoothing. [10] founds that additionally conducting non-linear activation and identity mapping for each layer on

APPNP further boosts the performance. On the other hand, the operation-based approaches develop certain operations upon the generic GCNs to prevent over-smoothing, such as the activation normalization [13], [14] and dropping edges [12], [11]. Our method, belonging to the operation-based family, delivers two enhancements of DropEdge and can be used along with the architecture-based methods.

III. PRELIMINARIES

Notations. We denote by $\mathcal{G} = (\mathbb{V}, \mathcal{E})$ the input graph of size N with nodes $v_i \in \mathbb{V}$ and edges $(v_i, v_j) \in \mathcal{E}$, by $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \in \mathbb{R}^{N \times C}$ the node features, and by $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ the adjacency matrix which associates edge (v_i, v_j) with $A(i, j)$. The node degrees $\boldsymbol{d} = \{d_1, \cdots, d_N\}$ compute the sum of edge weights connected to node i by d_i. The matrix form of $\boldsymbol{d}$ is defined as $\boldsymbol{D}$ whose diagonal elements are obtained from $\boldsymbol{d}$. The symbol "$\sim$" denotes random sampling.

GCN. As originally developed by [1], GCN recursively updates

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{1}$$

where $\boldsymbol{H}^{(l+1)} = \{\boldsymbol{h}_1^{(l+1)}, \cdots, \boldsymbol{h}_N^{(l+1)}\}$ are the hidden vectors of the l-th layer with $\boldsymbol{h}_i^{(l)}$ as the hidden feature for node i; $\hat{\boldsymbol{A}} = \hat{\boldsymbol{D}}^{-1/2}(\boldsymbol{A} + \boldsymbol{I})\hat{\boldsymbol{D}}^{-1/2}$ is the re-normalization of the adjacency matrix, and $\hat{\boldsymbol{D}}$ is the degree matrix of $\boldsymbol{A} + \boldsymbol{I}$; $\sigma(\cdot)$ is a nonlinear function, *i.e.* the ReLU function; and $\boldsymbol{W}^{(l)} \in \mathbb{R}^{C_l \times C_{l+1}}$ is the filter matrix in the l-th layer with C_l being the feature dimension. We assume C_l of all layers to be the same, $C_l = C$ in our following context unless otherwise specified.

In contrast to MLP, GCN by additionally performing message passing along $\hat{\boldsymbol{A}}$ is able to capture the topology of the input graph. Yet, this will unfortunately incur the over-smoothing issue for deep GCNs. We review the conclusion by [7] and repeat its compact form by [12] here.

Theorem 1 (Over-Smoothing[7]). *Let $\lambda_1 \leq \cdots \leq \lambda_N$ be the eigenvalues of $\hat{\boldsymbol{A}}$, sorted in ascending order. Suppose the multiplicity of the largest eigenvalue λ_N is $M(\leq N)$, i.e., $\lambda_{N-M} < \lambda_{N-M+1} = \cdots = \lambda_N$ and the second largest eigenvalue is defined as $\lambda := \max_{n=1}^{N-M} |\lambda_n|$, and let $\boldsymbol{E} \in \mathbb{R}^{N \times M}$ to be the eigenspace associated with $\lambda_{N-M+1}, \cdots, \lambda_N$. We denote the maximum singular value of $\boldsymbol{W}_l$ by s_l (assuming $s_l \leq 1$ here). Then we have $\lambda < \lambda_{N-M+1} = \cdots = \lambda_N = 1$, and*

$$d_{\mathcal{M}}(\boldsymbol{H}^{(l)}) \leq s_l \lambda d_{\mathcal{M}}(\boldsymbol{H}^{(l-1)}), \tag{2}$$

where $\mathcal{M} := \{\boldsymbol{E}\boldsymbol{C} | \boldsymbol{C} \in \mathbb{R}^{M \times C}\}$, and the Frobenius-norm-induced distance $d_{\mathcal{M}}(\boldsymbol{H}) := \inf_{\boldsymbol{Y} \in \mathcal{M}} ||\boldsymbol{H} - \boldsymbol{Y}||_{\mathrm{F}}$. That is to say, as $l \to \infty$, the output of GCN on $\mathcal{G}$ exponentially approaches a lower-dimensional space $\mathcal{M}$ that is independent to the input $\boldsymbol{X}$, incurring over-smoothing.

This theorem states that the spectral of $\hat{\boldsymbol{A}}$ has a crucial impact on over-smoothing. A potential way to alleviate over-smoothing (regardless of s_l that is affected by training) is by enlarging the value of λ to alleviate the condition $\lambda < 1$, which is the goal of DropEdge [12].
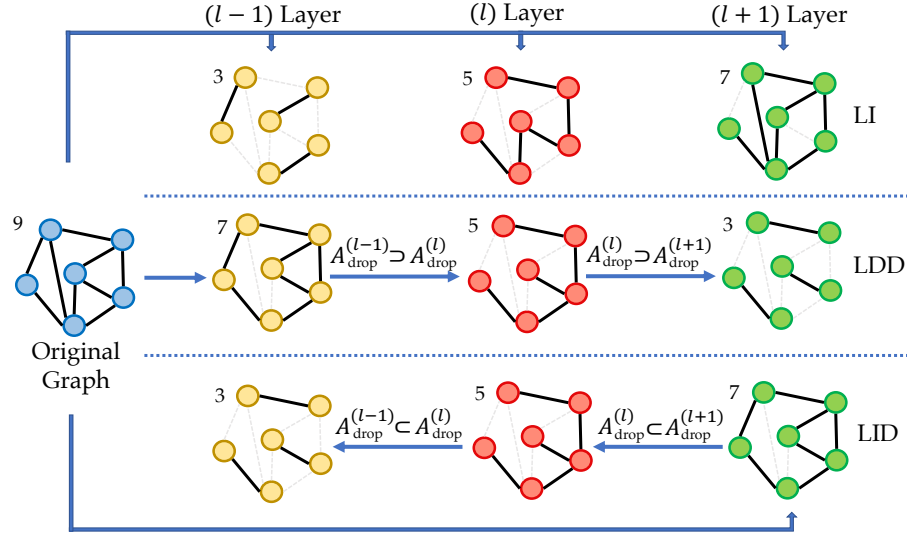
Fig. 1. LI, LDD and LID. Solid lines are the preserved edges. The number on the top left is $|\boldsymbol{A}_{\text{drop}}^{(i)}|$ for the current layer i.

DropEdge. [12] found that the number of edges is negatively related to the value of λ, and dropping edges to a certain extent will possibly increase the value of λ thus relieving over-smoothing. In practice, they try a random dropping process to prevent over-fitting as well. At each training iteration, DropEdge uniformly drops a subset of edges of a preset size $p|\mathcal{E}|$ by a non-replacement sampling process, *i.e.*,

$$\boldsymbol{A}_{\text{drop}} \sim \text{DropEdge}(\boldsymbol{A}, p|\mathcal{E}|), \tag{3}$$

which can be also represented in a probability form, $\boldsymbol{A}_{\text{drop}}(i,j) = \boldsymbol{A}(i,j) * \text{Bernoulli}(1-p)$. In practice, The re-normalization trick is further conducted on $\boldsymbol{A}_{\text{drop}}$, leading to $\hat{\boldsymbol{A}}_{\text{drop}}$. The authors also prove in theory the benefits of using DropEdge (see [12], [29] for more details).

IV. PROPOSED METHOD: DROPEDGE++

This section presents the details of DropEdge++, including the two enhanced samplers: Layer-Dependent (LD) sampler and Feature-Dependent (FD) sampler. We also provide essential theoretical justifications.

A. Layer-Dependent Sampler

DropEdge only sparsifies the adjacency matrix once and keeps it unchanged for all layers. One might be interested in if we can remove this constraint and adopt different adjacency matrices for different layers to involve more flexibility. Inspired by this, Layer-Independent (LI) DropEdge proposed by [12] constructs the different-layer adjacency matrix independently. However, LI is still unable to study how different-layer adjacency influences each other and what relation in-between helps prevent over-smoothing. This sub-section investigates these questions in the form of layer-dependent DropEdge. Prior to going further, we first reformulate GCN (Eq. (1)) as the following Layer-wise GCN (L-GCN):

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\boldsymbol{A}^{(l)}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{4}$$

where the adjacency matrix $\boldsymbol{A}^{(l)}$ varies for different layer. In LI [12], $\boldsymbol{A}^{(l)}$ is given by performing Eq. (3)

for each layer independently. Different from LI, we introduce two variants of layer-dependent methods below: Layer-Increasingly-Dependent (LID) DropEdge and Layer-Decreasingly-Dependent (LDD) DropEdge.

LID creates the adjacency matrix of the current layer by dropping a fixed number of edges conditional on the higher layer. In form, the adjacency matrix of the l-th layer is recursively computed by

$$\boldsymbol{A}_{\text{drop}}^{(l)} \sim \text{DropEdge}(\boldsymbol{A}_{\text{drop}}^{(l+1)}, \delta p|\mathcal{E}|), \tag{5}$$

where $\delta p = (p_{\max} - p_{\min})/(L-1)$, and $p_{\min}$ and $p_{\max}$ are hyper-parameters. For the top layer, $\boldsymbol{A}_{\text{drop}}^{(L-1)} \sim \text{DropEdge}(\boldsymbol{A}, p_{\min}|\mathcal{E}|)$. Obviously, we have $\boldsymbol{A}_{\text{drop}}^{(0)} \subset \boldsymbol{A}_{\text{drop}}^{(1)} \subset \cdots \subset \boldsymbol{A}_{\text{drop}}^{(L-1)}$, from sparse to dense.

LDD, in contrast to LID, derives the adjacency matrix of the current layer upon the lower layer,

$$\boldsymbol{A}_{\text{drop}}^{(l)} \sim \text{DropEdge}(\boldsymbol{A}_{\text{drop}}^{(l-1)}, \delta p|\mathcal{E}|), \tag{6}$$

and $\boldsymbol{A}_{\text{drop}}^{(0)} \sim \text{DropEdge}(\boldsymbol{A}, p_{\min}|\mathcal{E}|)$. Clearly, $\boldsymbol{A}_{\text{drop}}^{(0)} \supset \boldsymbol{A}_{\text{drop}}^{(1)} \supset \cdots \supset \boldsymbol{A}_{\text{drop}}^{(L-1)}$, from dense to sparse. Both LID and LDD perform adjacency normalization and compute $\boldsymbol{A}^{(l)} = \hat{\boldsymbol{A}}_{\text{drop}}^{(l)}$ for Eq. (4). The mechanisms of LI, LID, and LDD are depicted in Fig. 1.

Now, we characterize the theoretical property of LID and LDD. Note that it is no longer feasible to apply Theorem 1 for analyzing the over-smoothing of LID and LDD because $\boldsymbol{A}_{\text{drop}}^{(l)}$ is varying over layers and the subspace which GCN will converge to is unknown. To overcome this difficulty, we first revisit why over-smoothing hinders the training. We compute the gradient of the training loss $\mathcal{L}$ *w.r.t.* $\boldsymbol{W}^{(l)}$ in Eq. (4):

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(l)}} = (\boldsymbol{Z}^{(l)})^{\text{T}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}^{(l)}}, \tag{7}$$

where $\mathcal{L}$ is the training loss, $\boldsymbol{Z}^{(l)} = \boldsymbol{A}^{(l)}\boldsymbol{H}^{(l)}$ and $\boldsymbol{U}^{(l)} = \boldsymbol{A}^{(l)}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}$. If over-smoothing happens and $\boldsymbol{Z}^{(l)}$ locates in a low-dimension space $\mathcal{M}$, performing the inner-product $(\boldsymbol{Z}^{(l)})^{\text{T}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}^{(l)}}$ dramatically decreases the magnitude of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(l)}}$,
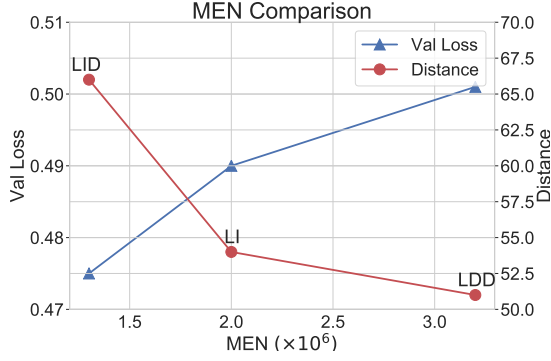
Fig. 2. MEN analysis upon 16-layer GCN on Cora. Distance is given by $\frac{1}{L-3}\sum_{l=1}^{L-3} D^{(l)}$, where $D^{(l)} = (\sum_{i,j}(\boldsymbol{H}_{ij}^{(l+1)} - \boldsymbol{H}_{ij}^{(l)})^2)^{\frac{1}{2}}$ returns the distance between the activations of adjacent layers. Obviously, less MEN derives larger Distance, implying less over-smoothing, hence smaller loss and better performance. More cases are shown in Fig. 3.

as projecting $\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}^{(l)}}$ into $\mathcal{M}$ will lose its values of most dimensions. It is thus difficult to train the model.

Next, we discuss the behavior of $\boldsymbol{Z}^{(l)}$ when $l \to \infty$. We approximate the analysis by relaxing the activation function to be linear, then the accumulated adjacency matrix from the input layer to the current layer becomes $\prod_{i=0}^{l} \boldsymbol{A}^{(i)}$. Based on the derivation in [12], the edge number of the adjacency matrix is positively related to the speed of over-smoothing as shown in Theorem 1. This motivates us to leverage the edge number of $\prod_{i=0}^{l} \boldsymbol{A}^{(i)}$ to measure the degree of over-smoothing for current layer $\boldsymbol{Z}^{(l)}$. Considering the updates of Eq. 7 for all layers, we compute the Mean Edge Number (MEN) of the accumulated adjacency matrices and introduce a new concept as follow.

Definition 1 (MEN). *Suppose $|\boldsymbol{A}|$ returns the edge number of $\boldsymbol{A}$. Then, the MEN of L-GCN is defined as MEN $= \frac{1}{L}\sum_{l=0}^{L-1} |\prod_{i=0}^{l} \boldsymbol{A}^{(i)}|$.*

Despite its approximation of linear activation in Eq. (4), MEN is able to provide us with a portable tool to theoretically investigate the convergence behavior of L-GCN given different edge samplers. More importantly, our experiments does verify that MEN is positively related to over-smoothing, and lower MEN yields better performance (see Fig. 2).

With the notion of MEN, we immediately have,

Theorem 2. *Let MEN_{lid} and MEN_{ldd} denote the expected MENs of the adjacency matrices sampled by LID and LDD, respectively, under the same values of p_{min} and p_{max}. Then, $MEN_{lid} \leq MEN_{ldd}$.*

We provide the full proof in Appendix A. It suggests that LID incurs the least MEN and delivers the best trainability and generalization than LDD. Note that Theorem 2 does not mean smaller MEN always leads to better performance. For instance, dropping all edges produces zero MEN and completely eliminates over-smoothing, but it yields performance detriment as well, since the model has degenerated to an MLP that is not topology-aware. Actually, we should choose edge sampling rate for each particular case to balance between reducing over-

smoothing and topology modeling. In our experiments, the best sampling rate is determined by validation.

LI is a layer-wise version of DropEdge, it is also valuable to justify how LID performs compared to LI. Yet it is not trivial to directly contrast their MENs in theory. Instead, we draw their correlation in terms of an upper-bound of MEN, defined as $MEN^* = \frac{1}{L}\sum_{l=0}^{L-1} \prod_{i=0}^{l} |\boldsymbol{A}^{(i)}|$.

Theorem 3. *Suppose LI shares the same dropping rate as LID for each layer, and let MEN_{lid}^* and MEN_{li}^* denote the expected MEN^* of LID and LI, respectively. Then, $MEN_{lid}^* \leq MEN_{li}^*$.*

B. Feature-Dependent Sampler

In this section, we introduce another sampler, FD. Unlike DropEdge in Eq. (3) where each edge is sampled uniformly, FD draws edges according to a measurement by a kernel function of node feature pairs. We define the kernel function as $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ that returns the similarity of two node features $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, for instance, $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x}_2$ in the linear case. Then, $\boldsymbol{A}_{\text{drop}}$ is constructed as follow,

$$\boldsymbol{A}_{\text{drop}} \sim \text{FD}(\boldsymbol{A}, \boldsymbol{X}, p|\mathcal{E}), \tag{8}$$

in other words,

$$\boldsymbol{A}_{\text{drop}}(i,j) = \boldsymbol{A}(i,j) * \text{Bernoulli}(\frac{(1-p)K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sum_{(i,j)\in\mathcal{E}} K(\boldsymbol{x}_i, \boldsymbol{x}_j)}) \tag{9}$$

Similar to DropEdge, we employ the non-replacement sampling and utilize the re-normalization trick on $\boldsymbol{A}_{\text{drop}}$ to obtain $\hat{\boldsymbol{A}}_{\text{drop}}$. The kernel computation is economical with the complexity of $O(|\mathcal{E}|)$. We pre-compute the kernels to eliminate the need of computing them at each training iteration in our implementation. The kernel here is applied to yield the sampling distribution for the edges, different from KerGNNs [21] which leverage graph kernels to compute the convolution between the local subgraph and the trainable graph filters.

Intuitively, the edges connecting two nodes of similar features are more potentially intra-class connections and should be preserved more often, while the edges connecting dissimilar nodes should be possibly removed as they are likely linking nodes of distinct classes. Our FD sampler does serve this goal by employing the kernel function as an additional weight of the edge sampling probability. In our experiments (§ V-D), FD outperforms all other counterparts, which verifies the validity of our design.

Furthermore, FD correlates the adjacency matrix with the node features. This is beneficial since the output of GCN still involves the information of the input features even if the over-smoothing is unavoidable, according to Theorem 1. We summarize this property as a theorem for better readability.

Theorem 4. *FD correlates the subspace $\mathcal{M}$ with the input features $\boldsymbol{X}$. To be specific, the bases of $\mathcal{M}$ are returned as $\boldsymbol{e}_m = \boldsymbol{D}^{\frac{1}{2}}\boldsymbol{u}_m$ for $m = 1, \cdots, M$, where the i-th expected diagonal element of $\boldsymbol{D}$ is $\mathbb{E}[D_i] = (1-p)\sum_j \frac{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{A}(i,j)}{\sum_{(i,j)\in\mathcal{E}} K(\boldsymbol{x}_i, \boldsymbol{x}_j)}$, and $\boldsymbol{u}_m$ is the indicator vector of the m-th connected component, i.e. $u_m(i) = 1$ if node i belongs to m and $u_m(i) = 0$ otherwise.*

The full details of the proof are included in Appendix B. By Theorem 4, our FD sampler can bridge the dependence between the convergence subspace and the input features, as opposed to DropEdge. This better relieves the effect of over-smoothing since the output after an infinite number of layers is still related to the input features.

Overall Framework. Our final version of DropEdge++, given the analyses above, is the integration of LID and FD. Specifically, we employ Eq. (5) but using $\boldsymbol{A}_{\text{drop}}^{(l)} \sim$ FD$(\boldsymbol{A}_{\text{drop}}^{(l+1)}, \boldsymbol{X}, \delta p|\mathcal{E}|)$. Note that Theorem 2 still holds under the FD sampler.

V. EXPERIMENTS

A. Datasets

To demonstrate the generality of our proposed method, we evaluate DropEdge++ on six node classification benchmarks, including both full- and semi- supervised tasks.

Citation Networks The Cora, Citeseer, and Pubmed [30] datasets are popular benchmarks for graph neural networks. The nodes represent papers while features are the bag-of-words representation of paper titles. Edges stand for the citation relationship between the papers, and the papers are labeled by the categories they belong to. On these three datasets, we leverage both the semi-supervised fashion according to the standard split by [31], [20], [1], [10] and the full-supervised mode following [26], [27], [12].

Co-author and Co-purchase Networks The Co-author CS and Co-author Physics are co-author networks [32] based on the Microsoft Academic Graph with nodes representing authors and edges representing the co-authorship on a paper. Features are constructed by the keywords for the authors' papers, and labels reflect the authors' active fields. The Amazon Photos [32] dataset is derived from Amazon co-purchase graph. Nodes represent goods, and edges represent close co-purchase relationship. Features are the product reviews in bag-of-words form, and labels indicate the category. Among these three datasets, we apply the semi-supervised tasks. We randomly fix 20 nodes per class for training, 30 nodes per class for validation, and all the rest for testing, following [11], [33].

B. Backbones

We implement DropEdge++ on 4 popular backbones: generic GCN [1], ResGCN [5], JKNet [8], and APPNP* that adds non-linearity to the propagation of APPNP [9]. We list the four propagation models below. For ResGCN [5], the residual connections within layers are implemented as the adding operation. For JKNet [8], the final output embedding is derived from the concatenation of the hidden layer outputs. APPNP* is an extension of APPNP [9] by adding non-linearity on the propagation model. Note that in Table II, the symbol || stands for concatenation. A similar model to APPNP* has been discussed by [10], where the authors found that adding both non-linearity and identity mapping to APPNP results in more desired performance for deep architectures. However, the identity mapping requires careful hyper-parameter tuning and we thus only apply non-linearity for simplicity and find it works well in our experiments.

Self-feature Modeling We also implement self-feature modeling [34] , which is a variant on the basic GCN propagation.

$$\boldsymbol{H}^{(l+1)} = \sigma \left(\hat{\boldsymbol{A}} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)} + \boldsymbol{H}^{(l)} \boldsymbol{W}_{\text{self}}^{(l)} \right) \qquad (10)$$

We can simply replace the convolution operation in GCN with Eq. (10) to obtain the self-feature modeling variant, which indeed could be well treated as a hyper-parameter for the potential enhancement on GCN, ResGCN, and JKNet.

C. How does LID act compared with other variants?

We start by validating how the proposed metric MEN affects the extent of over-smoothing. Fig. 3 shows the correlation between MEN and the validation loss as well as the average distance of the hidden-layer outputs. The experiments are conducted on GCN with various layers on Cora dataset, with the four comparing sampling methods (LID, LDD, LI, and DropEdge). The results consistently show that MEN is negatively correlated with the average distance and positively related to the validation loss. In general, a lower MEN yields less over-smoothing given the same dropping rate, which aligns with our theory. Moreover, consistent with Theorem 2 and Theorem 3, LID produces the lowest MEN and validation loss among all compared candidates. Although it is non-trivial to compare MEN_{lid} and MEN_{li} theoretically, in practice we find that $\text{MEN}_{\text{lid}} \leq \text{MEN}_{\text{li}}$ holds based on our observation. More discussions on the comparison of MENs are provided in Appendix E-B.

We then give a detailed comparison in terms of the training and validation curves. Fig. 4 displays the training curves and validation curves for DropEdge, LDD, LID, and LI on a 6-layer GCN under the same average dropping rate. Clearly, DropEdge gains inferior performance, indicating that employing shared adjacency matrices in different layers is not a good choice in this case. Particularly, LID achieves lower training and validation losses than LDD and LI, which verifies its better capability of preventing over-smoothing.

D. How does FD act compared with DropEdge?

Theorem 4 has proved that FD improves the expressivity of deep GCNs by associating the convergence subspace with the input features. To justify if this is the case in practice, we carry out FD (with the Linear kernel) and DropEdge upon 6-layer GCN under the same dropping rate. Apparently in Fig. 5, our FD sampler outperforms DropEdge in terms of both the training and validation curves. As an oracle reference, we also involve the loss of DropEdge++ that is equipped with FD and LID. DropEdge++ achieves the lowest loss than other variants, showing the compatibility of FD and LID. Except for the Linear kernel, we also test other kinds of kernels including the Polynomial kernel, RBF kernel for FD, which are all positively correlated with the feature-similarity. The specifications of these kernels are deferred to Appendix D-B. To further validate our design, we also include a reverse version of FD, *i.e.*, $K_{\text{Rev}} = 1 - K_{\text{Linear}}$. The results in the right sub-figure of Fig. 5 indicates that all kernels are better than DropEdge, while

TABLE I
DATASETS STATISTICS

Dataset	Node	Edge	Feature	Class	Training / Validation / Testing
Cora (Full)	2,708	5,429	1,433	7	1,208 / 500 / 1,000
Citeseer (Full)	3,327	4,732	3,703	6	1,812 / 500 / 1,000
Pubmed (Full)	19,717	44,338	500	3	18,217 / 500 / 1,000
Cora (Semi)	2,708	5,429	1,433	7	140 / 500 / 1,000
Citeseer (Semi)	3,327	4,732	3,703	6	120 / 500 / 1,000
Pubmed (Semi)	19,717	44,338	500	3	60 / 500 / 1,000
Coauthor CS	18,333	81,894	6,805	15	300 / 450 / 17,583
Coauthor Physics	34,493	247,962	8,415	5	100 / 150 / 34,243
Amazon Photos	7,487	119,043	745	8	160 / 240 / 7,087
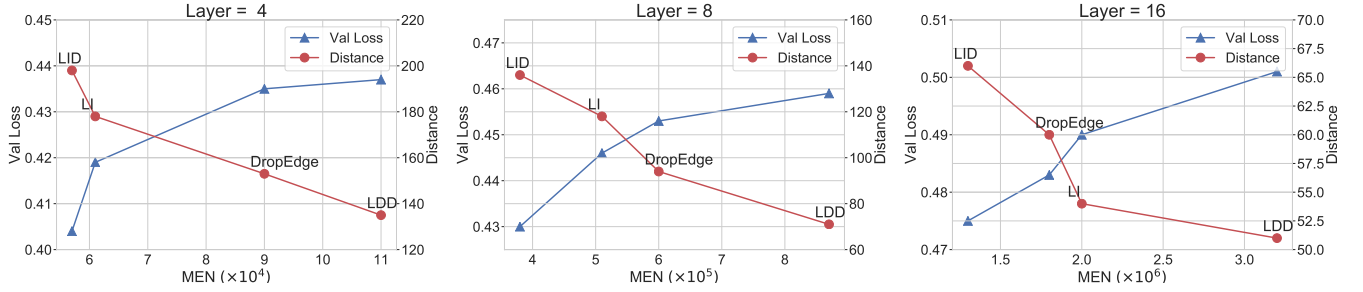
Fig. 3. MEN vs Validation Loss and Average Distance on full-supervised Cora, a more complete version of Fig. 2.
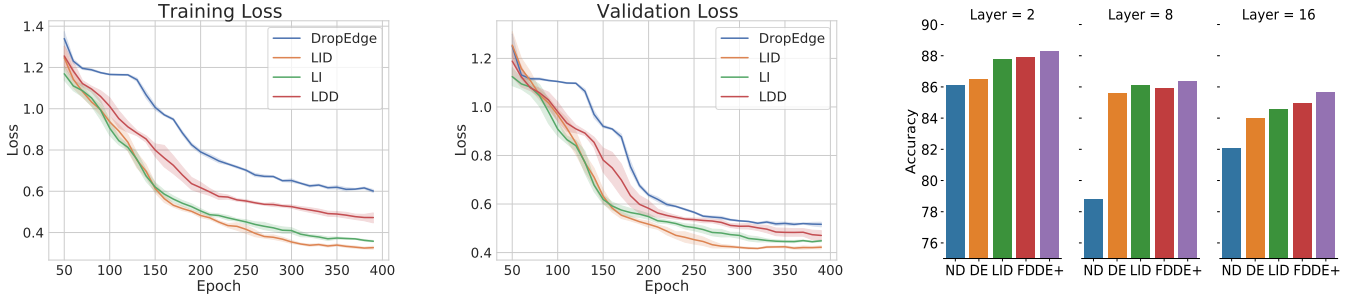
Fig. 4. Analysis of LID on full-supervised Cora. (1) Left: training loss; Middle: validation loss. LID, LDD and LI share the same sampling rate p_{max} and p_{min}, while the rate of DropEdge is set to be $(p_{max} + p_{min})/2$ for fair comparison. Colored area indicates one standard deviation over 10 runs. (2) Right: Accuracy comparison of different variants (ND: NoDrop, DE: DropEdge, DE+: DropEdge++).
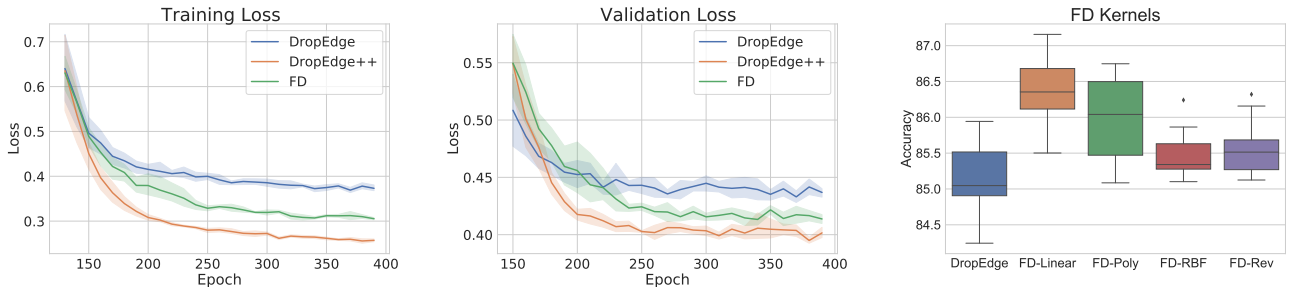
Fig. 5. Analysis of FD on full-supervised Cora. Left: training loss; Middle: validation loss; Right: FD with various kernels. FD is implemented with the Linear kernel by default.

Polynomial and Linear are most desirable in this case. We will use the Linear kernel in what follows by default.

As a more complete ablation study, the right sub-figure in Fig. 4 compares NoDrop, DropEdge++, and its two components, LID and FD. In accordance with our analysis, both FD and LID improve the performance over the generic GCN as well as DropEdge, while DropEdge++ achieves the best performance with the help of its two structure-aware samplers.

TABLE II
THE PROPAGATION MODELS.

Backbone	Propagation
GCN	$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right)$
ResGCN	$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right) + \boldsymbol{H}^{(l)}$
JKNet	$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right),$ $\boldsymbol{H} = \|\|_{l=0}^{L}\boldsymbol{H}^{(l)}$
APPNP*	$\boldsymbol{Z}^{(0)} = f_\theta(\boldsymbol{X}),$ $\boldsymbol{Z}^{(l+1)} = \sigma((1-\alpha)\hat{\boldsymbol{A}}^{(l)}\boldsymbol{Z}^{(l)} + \alpha\boldsymbol{Z}^{(0)})$

TABLE III
TESTING ACCURACY (%) ON DIFFERENT BACKBONES
(SEMI-SUPERVISED). ND: NODROP, DE: DROPEDGE, DE+:
DROPEDGE++.

	Layer		2	4	8	16	32	64
Cora	GCN	ND	81.1	80.4	69.5	64.9	60.3	28.7
		DE	82.8	82.0	75.8	75.7	62.5	49.5
		DE+	**82.9**	**83.1**	**80.0**	**77.3**	**71.7**	**55.2**
	ResGCN	ND	-	78.8	75.6	72.2	76.6	61.1
		DE	-	**83.3**	82.8	82.7	81.1	78.9
		DE+	-	83.2	**83.2**	**82.7**	**82.8**	**82.6**
	JKNet	ND	-	80.2	80.7	80.2	81.1	71.5
		DE	-	83.3	82.6	83.0	82.5	83.2
		DE+	-	**83.4**	**83.4**	**83.6**	**83.7**	**83.8**
	APPNP*	ND	-	83.2	82.9	82.7	83.3	82.2
		DE	-	82.5	82.3	82.9	83.0	82.4
		DE+	-	**84.6**	**85.0**	**84.8**	**84.7**	**84.9**
Citeseer	GCN	ND	70.8	67.6	30.2	18.3	25.0	20.0
		DE	72.3	70.6	61.4	57.2	41.6	34.4
		DE+	**72.7**	**72.3**	**68.9**	**64.2**	**50.1**	**37.0**
	ResGCN	ND	-	70.5	65.0	66.5	62.6	22.1
		DE	-	72.2	71.6	70.1	70.0	65.1
		DE+	-	**72.9**	**73.1**	**73.3**	**73.0**	**73.6**
	JKNet	ND	-	68.7	67.7	69.8	68.2	63.4
		DE	-	72.6	71.8	72.6	70.8	72.2
		DE+	-	**73.7**	**74.1**	**73.9**	**73.6**	**73.3**
	APPNP*	ND	-	72.3	71.8	73.1	72.7	72.1
		DE	-	72.4	72.5	72.3	73.5	72.7
		DE+	-	**74.5**	**74.8**	**74.8**	**74.9**	**74.7**
Pubmed	GCN	ND	79.0	76.5	61.2	40.9	22.4	35.3
		DE	79.6	79.4	78.1	78.5	77.0	61.5
		DE+	**79.8**	**80.0**	**79.9**	**78.8**	**77.5**	**74.7**
	ResGCN	ND	-	78.6	78.1	75.5	67.9	66.9
		DE	-	78.8	78.9	78.0	78.2	76.9
		DE+	-	**80.2**	**80.6**	**80.1**	**79.9**	**79.9**
	JKNet	ND	-	78.0	78.1	72.6	72.4	74.5
		DE	-	78.7	78.7	79.1	79.2	78.9
		DE+	-	**80.0**	**80.2**	**80.3**	**80.4**	**80.0**
	APPNP*	ND	-	79.5	79.9	79.8	80.2	80.3
		DE	-	80.1	80.4	80.3	80.0	80.5
		DE+	-	**80.9**	**81.0**	**81.3**	**81.6**	**81.8**

TABLE IV
TESTING ACCURACY (%) COMPARISON ON DIFFERENT BACKBONES
(FULL-SUPERVISED).

	Layer		2	4	8	16	32	64
Cora	GCN	ND	86.1	85.5	78.8	82.1	71.6	52.0
		DE	86.5	87.6	85.8	84.3	74.6	53.2
		DE+	**88.3**	**87.8**	**86.4**	**85.7**	**82.9**	**74.9**
	ResGCN	ND	-	86.0	85.4	85.3	85.1	79.8
		DE	-	87.0	86.9	86.9	86.8	84.8
		DE+	-	**88.5**	**88.4**	**88.2**	**88.0**	**87.9**
	JKNet	ND	-	86.9	86.7	86.2	87.1	86.3
		DE	-	87.7	87.8	88.0	87.6	87.9
		DE+	-	**87.8**	**88.3**	**88.6**	**88.1**	**88.2**
	APPNP*	ND	-	88.3	88.5	88.5	88.4	88.5
		DE	-	87.4	88.4	88.3	88.4	88.2
		DE+	-	**88.7**	**89.0**	**89.0**	**89.2**	**89.1**
Citeseer	GCN	ND	75.9	76.7	74.6	65.2	59.2	44.6
		DE	78.7	79.2	77.2	76.8	61.4	45.6
		DE+	**80.5**	**79.5**	**78.3**	**77.2**	**75.2**	**56.5**
	ResGCN	ND	-	78.9	77.8	78.2	74.4	21.2
		DE	-	78.8	78.8	79.4	77.9	75.3
		DE+	-	**80.6**	**80.8**	**81.0**	**80.3**	**80.2**
	JKNet	ND	-	79.1	79.2	78.8	71.7	76.7
		DE	-	80.2	80.2	80.1	80.0	80.0
		DE+	-	**81.1**	**81.0**	**80.5**	**80.2**	**80.8**
	APPNP*	ND	-	79.3	80.6	80.6	79.0	79.5
		DE	-	79.5	80.3	79.9	79.5	79.6
		DE+	-	**80.9**	**81.0**	**81.2**	**81.2**	**81.1**
Pubmed	GCN	ND	90.2	88.7	90.1	88.1	84.6	79.7
		DE	91.2	91.3	90.9	90.3	86.2	79.0
		DE+	**91.8**	**91.0**	**91.0**	**90.7**	**90.5**	**83.1**
	ResGCN	ND	-	90.7	89.6	89.6	90.2	87.9
		DE	-	90.7	90.5	91.0	91.1	90.2
		DE+	-	**91.0**	**91.6**	**91.3**	**91.7**	**91.2**
	JKNet	ND	-	90.5	90.6	89.9	89.2	90.6
		DE	-	**91.3**	91.2	91.5	91.3	**91.6**
		DE+	-	91.2	**91.6**	**91.6**	91.5	**91.6**
	APPNP*	ND	-	90.5	90.3	90.6	90.5	90.2
		DE	-	90.4	90.6	90.5	90.6	90.2
		DE+	-	**90.9**	**91.2**	**91.0**	**90.9**	**90.9**

E. The performance of DropEdge++ on various backbones of different layers.

We compare the testing accuracy of NoDrop, DropEdge and DropEdge++ with various layers (from 2 to 64) and backbones. More details about the design of hyper-parameters (such as $p_{\min}$ and $p_{\max}$) are provided in Appendix D-A.

Table III and Table IV display the results on Cora, Citeseer and Pubmed under semi- and full- supervised settings, respectively. We reuse the best metrics reported in [12] for NoDrop and DropEdge. The results on the other three datasets are deferred to Appendix E-A.

We have these findings:

1. It is clearly shown that DropEdge++ is consistently effective in boosting the performance over NoDrop and DropEdge on all backbones with varying layers.

2. As for GCN, even equipped with DropEdge++, deep models are generally worse than the shallow counterparts under the full-supervised setting. Even so, DropEdge++, as a robust sampling technique, still enhances the performance of the 2-layer models over NoDrop remarkably; for example, the accuracy is increased from 86.1 to 88.3 on full-supervised Cora. Moreover, when deep GCN fails and delivers much worse performance, conducting DropEdge++ yields promising

TABLE V
TESTING ACCURACY (%) UNDER SEMI-SUPERVISED SETTINGS. THE NUMBER IN PARENTHESIS DENOTES THE DEPTH.

	Cora	Citeseer	Pubmed	Coauthor CS	Coauthor Phy.	Amazon Photos
GCN	81.5	71.1	79.0	89.80	92.80	90.60
GAT	83.1	70.8	78.5	90.89	91.10	89.70
APPNP	83.3	71.8	80.1	92.08	93.32	91.42
GCNII	**85.5**	73.4	80.2	91.34	93.43	91.05
GCN (Drop)	82.8	72.3	79.6	91.90	93.23	91.58
JKNet	81.1	69.8	78.1	92.20	92.94	91.69
GCN (AdaEdge)	82.3	69.7	77.4	90.30	93.00	91.50
GCN w DE+	$83.1_{\pm0.2(4)}$	$72.8_{\pm0.4(2)}$	$80.0_{\pm0.3(4)}$	$92.96_{\pm0.04(2)}$	$\mathbf{94.45}_{\pm0.10(4)}$	$92.22_{\pm0.42(4)}$
ResGCN w DE+	$83.2_{\pm0.4(4)}$	$73.6_{\pm0.3(64)}$	$80.6_{\pm0.5(8)}$	$92.88_{\pm0.18(16)}$	$94.04_{\pm0.11(16)}$	$\mathbf{93.06}_{\pm0.28(4)}$
JKNet w DE+	$83.8_{\pm0.4(64)}$	$74.1_{\pm0.5(8)}$	$80.4_{\pm0.2(32)}$	$93.05_{\pm0.29(8)}$	$94.21_{\pm0.16(8)}$	$92.94_{\pm0.26(8)}$
APPNP* w DE+	$85.0_{\pm0.3(8)}$	$\mathbf{74.9}_{\pm0.4(32)}$	$\mathbf{81.8}_{\pm0.3(64)}$	$\mathbf{93.10}_{\pm0.15(8)}$	$94.42_{\pm0.22(16)}$	$93.01_{\pm0.21(4)}$

TABLE VI
TESTING ACCURACY (%) UNDER FULL-SUPERVISED SETTINGS. THE NUMBER IN PARENTHESIS DENOTES THE DEPTH.

	Cora	Citeseer	Pubmed
GCN	86.64	79.34	90.22
GAT	87.38	78.60	89.66
FastGCN	85.00	77.60	88.00
AS-GCN	87.44	79.66	90.60
APPNP	87.87	79.82	90.36
GCNII	88.25	79.92	90.05
GCN (Drop)	87.60	79.20	91.30
GraphSAGE	82.20	71.40	87.10
GCN w DE+	$88.30_{\pm0.16(2)}$	$80.52_{\pm0.14(2)}$	$\mathbf{91.76}_{\pm0.16(2)}$
ResGCN w DE+	$88.48_{\pm0.32(4)}$	$80.95_{\pm0.24(16)}$	$91.66_{\pm0.22(32)}$
JKNet w DE+	$88.55_{\pm0.15(16)}$	$81.06_{\pm0.40(4)}$	$91.57_{\pm0.31(8)}$
APPNP* w DE+	$\mathbf{89.22}_{\pm0.11(32)}$	$\mathbf{81.21}_{\pm0.19(16)}$	$91.20_{\pm0.06(8)}$

results over NoDrop and DropEdge, *e.g.* 74.9 vs 53.2 with 64-layer GCN on full-supervised Cora.

3. When using ResGCN, JKNet or APPNP* as the backbone, DropEdge++ generally holds steady with the increase of network depth and nearly gains the best number when the depth is beyond 4. For instance, when equipped with DropEdge++, the 8-layer JKNet achieves 74.1 and 32-layer APPNP* achieves 74.9 on semi-supervised Citeseer.

F. Comparison with SOTAs

We choose the best performance for each backbone with DropEdge++, and contrast them with State of the Arts (SOTAs) in Table V and Table VI, including GCN, GCN with DropEdge, GAT, AS-GCN, FastGCN, APPNP, GraphSAGE, GCNII, and AdaEdge [11]. We reuse the results of SOTAs reported in [12], [10] on Cora, Citeseer and Pubmed, and those on the other three datasets from [11], [35], [33]. The means and standard deviations are acquired through 10 runs, with detailed hyper-parameters in Appendix D-A. Obviously, our DropEdge++ obtains enhancement against SOTAs with an average improvement of more than 1% on nearly all the benchmarks, which is significant given the challenge of these benchmarks. Although the performance on semi-supervised Cora is worse than GCNII possibly due to limited training

labels, DropEdge++ outperforms all SOTAs on full-supervised Cora. Moreover, the best accuracy for most models with DropEdge++ is attained under the depth beyond 2, which again verifies the impact of DropEdge++ on formulating deep networks.

G. Time Complexity

Although we adopt a layer-wise sampling approach, the time consumption of DropEdge++ is not heavily burdened compared with DropEdge, since the kernel matrix in FD is pre-computed before training. As displayed in Table VII with GCN on Cora dataset, it only occupies a small percentage of the overall running time for sampling, making our DropEdge++ competitively efficient as DropEdge.

TABLE VII
AVERAGE TIME CONSUMED PER EPOCH ON CORA (IN SECONDS).

Method	NoDrop	DropEdge	DropEdge++
Sampling	0.0015	0.0022	0.0043
Training	0.0125	0.0115	0.0117
Total	0.0140	0.0137	0.0160

VI. CONCLUSION

In this paper, we investigate over-smoothing on GCNs and develop two effective samplers, *i.e.* the layer-increasingly-dependent sampler and the feature-dependent sampler. The layer-increasingly-dependent sampler is able to decrease the metric, mean-edge-number (MEN), and thus further reduces over-smoothing while the feature-dependent sampler further improves the expressivity of deep GCNs. Our overall framework DropEdge++ integrates these two structure-aware samplers and is demonstrated to be effective on various backbones of both deep and shallow models towards both full- and semi-supervised node classification.

APPENDIX A
PROOF OF THEOREM 2

Recall that by definition, the MEN of L-GCN is computed as $\text{MEN} = \frac{1}{L} \sum_{l=0}^{L-1} |\prod_{i=0}^{l} \boldsymbol{A}^{(i)}|$, and we have the following theorem.

Theorem 2. *Let MEN_{lid} and MEN_{ldd} denote the expected MENs of the adjacency matrices sampled by LID and LDD, respectively, under the same values of p_{min} and p_{max}. Then, $\text{MEN}_{lid} \leq \text{MEN}_{ldd}$.*

Proof. Without loss of generality, we assume the adjacency matrices sampled by LID and LDD are in the reverse order to each other, namely,

$$
\begin{aligned}
\mathbb{A}_{\text{lid}} &= \{\boldsymbol{A}_0, \boldsymbol{A}_1, ..., \boldsymbol{A}_{L-1}\}, \\
\mathbb{A}_{\text{ldd}} &= \{\boldsymbol{A}_{L-1}, \boldsymbol{A}_{L-2}, ..., \boldsymbol{A}_0\},
\end{aligned}
$$

where $\boldsymbol{A}_0 \subset \boldsymbol{A}_1 \subset ... \subset \boldsymbol{A}_{L-1}$. Further, we denote the accumulated products as $\boldsymbol{A}'_l = \prod_{i=0}^{l} \boldsymbol{A}_i$ and $\boldsymbol{A}''_l = \prod_{i=L-1}^{L-l-1} \boldsymbol{A}_i$. By these denotations, we arrive at

$$
\begin{aligned}
\text{MEN}_{\text{lid}} &= \frac{1}{L} \sum_{l=0}^{L-1} |\boldsymbol{A}'_l|, \\
\text{MEN}_{\text{ldd}} &= \frac{1}{L} \sum_{l=0}^{L-1} |\boldsymbol{A}''_l|.
\end{aligned}
$$

In order to prove $\text{MEN}_{\text{lid}} \leq \text{MEN}_{\text{ldd}}$, we prove $|\boldsymbol{A}'_l| \leq |\boldsymbol{A}''_l|$ for each $l \in [0, L-1]$.

Note that $\boldsymbol{A}_{i-L+l+1} \subset \boldsymbol{A}_i$ since $i - L + l + 1 \leq i$, for $0 \leq l \leq L - 1$. So, we can define $\boldsymbol{A}_i = \boldsymbol{A}_{i-L+l+1} + \delta\boldsymbol{A}_{i-L+l+1}$ for abbreviation, where $\delta\boldsymbol{A}_{i-L+l+1} \geq 0$. Putting this into $\boldsymbol{A}''_l$, we easily devise

$$
\begin{aligned}
\boldsymbol{A}''_l &= \prod_{i=L-1}^{L-l-1} \boldsymbol{A}_i, \\
&= \prod_{i=L-1}^{L-l-1} (\boldsymbol{A}_{i-L+l+1} + \delta\boldsymbol{A}_{i-L+l+1}), \\
&= \boldsymbol{B} + \prod_{i=L-1}^{L-l-1} \boldsymbol{A}_{i-L+l+1}, \\
&= \boldsymbol{B} + \prod_{i=l}^{0} \boldsymbol{A}_i, \\
&= \boldsymbol{B} + (\boldsymbol{A}'_l)^{\text{T}},
\end{aligned}
$$

where $\boldsymbol{B} \geq 0$, and the last equivalence holds due to the symmetry of each adjacency matrix. Hence, it is natural to see that $|\boldsymbol{A}'_l| \leq |\boldsymbol{A}''_l|$, which concludes the proof. $\square$

APPENDIX B
PROOF OF THEOREM 3

In theory we find it non-trivial to directly compare the MENs of LID and its layer-independent counterpart LI if they share exactly the same dropping rate per layer. To bypass this difficulty, we investigate the relationship by comparing an upper-bound of the MEN, defined as $\text{MEN}^* = \frac{1}{L} \sum_{l=0}^{L-1} \prod_{i=0}^{l} |\boldsymbol{A}^{(i)}|$.

For MEN^* and MEN, we always have $\text{MEN}^* \geq \text{MEN}$, because $|\boldsymbol{A}||\boldsymbol{B}| \geq |\boldsymbol{AB}|$ holds for all $\boldsymbol{A}, \boldsymbol{B}$ with non-negative entries, and it concludes naturally by induction.

Theorem 3. *Suppose LI shares the same dropping rate as LID for each layer, and let MEN^*_{lid} and MEN^*_{li} denote the expected MEN^* of LID and LI, respectively. Then, $\text{MEN}^*_{lid} \leq \text{MEN}^*_{li}$.*

Proof. Denote $P(m_{l-1}|m_l)$ as the probability of the edge number of the $(l-1)$-th layer being m_{l-1}, conditioned on the edge number of the l-th layer being m_l. Further denote $|\boldsymbol{A}| = n$, and the rate of preserved edges for the l-th layer as p_l.

$$
\begin{aligned}
&\mathbb{E}_{\text{lid}} \left(\prod_{i=0}^{l} |\boldsymbol{A}_{\text{drop}}^{(i)}| \right) \\
&= \sum_{m_0 \leq m_1 \leq ... \leq m_l} \left(P(m_l) \prod_{i=0}^{l-1} P(m_{l-i-1}|m_{l-i}) \prod_{i=0}^{l} m_i \right), \\
&= \sum_{m_1 \leq ... \leq m_l} \left(P(m_l) \prod_{i=0}^{l-2} P(m_{l-i-1}|m_{l-i}) \prod_{i=1}^{l} m_i \right) \\
&\quad \cdot \sum_{m_0 \leq m_1} P(m_0|m_1) m_0, \\
&= \sum_{m_1 \leq ... \leq m_l} \left(P(m_l) \prod_{i=0}^{l-2} P(m_{l-i-1}|m_{l-i}) \prod_{i=1}^{l} m_i \right) p_0 m_1, \\
&\leq \sum_{m_1 \leq ... \leq m_l} \left(P(m_l) \prod_{i=0}^{l-2} P(m_{l-i-1}|m_{l-i}) \prod_{i=1}^{l} m_i \right) p_0 n, \\
&\leq \cdots \\
&\leq \prod_{i=0}^{l} (p_i n), \\
&= \mathbb{E}_{\text{li}} \left(\prod_{i=0}^{l} |\boldsymbol{A}_{\text{drop}}^{(i)}| \right).
\end{aligned}
$$

Hence, we have $\text{MEN}^*_{\text{lid}} \leq \text{MEN}^*_{\text{li}}$.

$\square$

APPENDIX C
PROOF OF THEOREM 4

Theorem 4. *FD correlates the subspace $\mathcal{M}$ with the input features $\boldsymbol{X}$. To be specific, the bases of $\mathcal{M}$ are returned as $e_m = \boldsymbol{D}^{\frac{1}{2}} \boldsymbol{u}_m$ for $m = 1, \cdots, M$, where the i-th expected diagonal element of $\boldsymbol{D}$ is $\mathbb{E}[D_i] = (1-p) \sum_j \frac{K(\boldsymbol{x}_i, \boldsymbol{x}_j) \boldsymbol{A}(i,j)}{\sum_{(i,j) \in \mathcal{E}} K(\boldsymbol{x}_i, \boldsymbol{x}_j)}$, and $\boldsymbol{u}_m$ is the indicator vector of the m-th connected component, i.e. $u_m(i) = 1$ if node i belongs to m and $u_m(i) = 0$ otherwise.*

TABLE VIII
HYPER-PARAMETER DESCRIPTIONS.

Hyper-parameter	Description
lr	the learning rate
L_2	L_2 regularization
$p_{\min}, p'_{\max}$	the min/max sampling rate
hidden	the dimension of hidden units
α	the coefficient in APPNP*
dropout	the dropout rate
normalization	the normalization method in Table IX
Loop	self-feature modeling
BN	batch normalization

TABLE IX
THE APPLIED NORMALIZATION METHODS.

Abbr.	Name and Mathematical Form
NA	Normlized Adjacency $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$
FOG	First-order GCN $\boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$
AN	Augmented Normalized Adjacency $(\boldsymbol{D}+\boldsymbol{I})^{-1/2}(\boldsymbol{A}+\boldsymbol{I})(\boldsymbol{D}+\boldsymbol{I})^{-1/2}$
ANS	Augmented Normalized Adjacency with Self-Loop $\boldsymbol{I} + (\boldsymbol{D}+\boldsymbol{I})^{-1/2}(\boldsymbol{A}+\boldsymbol{I})(\boldsymbol{D}+\boldsymbol{I})^{-1/2}$
ARW	Augmented Random Walk $(\boldsymbol{D}+\boldsymbol{I})^{-1}(\boldsymbol{A}+\boldsymbol{I})$

Proof. We borrow Proposition 1 [7] and find that the convergence subspace $\mathcal{M}$ of GCN is formulated by the bases of $\boldsymbol{e}_m = \boldsymbol{D}^{\frac{1}{2}}\boldsymbol{u}_m$ for $m = 1, \cdots, M$ where $\boldsymbol{D}$ is the degree matrix of the normalized adjacency matrix of the input graph $\mathcal{G}$, and $\boldsymbol{u}_m$ is the indicator vector of the m-th connected component of $\mathcal{G}$, *i.e.* $u_m(i) = 1$ if node i belongs to m and $u_m(i) = 0$ otherwise. As it also holds for the graph $\mathcal{G}$ after performing FD, the bases of convergence subspace $\mathcal{M}$ become $\boldsymbol{e}_m = \boldsymbol{D}^{\frac{1}{2}}\boldsymbol{u}_m$ for $m = 1, \cdots, M$. It is natural to see that $\mathbb{E}[D_i] = \mathbb{E}[\sum_j \boldsymbol{A}_{\mathrm{drop}}(i,j)] = (1-p)\sum_j \frac{K(\boldsymbol{x}_i,\boldsymbol{x}_j)\boldsymbol{A}(i,j)}{\sum_{(i,j)\in\mathcal{E}} K(\boldsymbol{x}_i,\boldsymbol{x}_j)}$. $\square$

APPENDIX D
EXPERIMENT DETAILS

A. Hyper-parameters

The hyper-parameters are listed in Table VIII. Our proposed DropEdge++ implements the LID sampler, which indicates the inequality $p_{\max} \geq p_{\min}$ always holds. Consequently, it is intuitive to denote $p'_{\max}$ as a replacement of $p_{\max}$ that satisfies $p'_{\max} = (p_{\max} - p_{\min})/(1 - p_{\min})$. With this transformation, the domain of $p_{\max}$ is modified into $0 \leq p'_{\max} \leq 1$ naturally in the LID sampler. In our experiments, we use $p_{\min}$ and $p'_{\max}$ to determine the sampling rates throughout the layers, with $0 \leq p_{\min} \leq 1$ and $0 \leq p'_{\max} \leq 1$.

The adopted normalization methods are presented in Table IX together with their abbreviations. Table X and Table XI show the values of hyper-parameters for achieving the best testing accuracy, chosen by a random search. Note that *norm.* and *drop.* are short for normalization, and dropout, respectively. Experiments are conducted with layer $\in \{2, 4, 8, 16, 32, 64\}$ for the citation networks and layer $\in \{2, 4, 8, 16, 32\}$ for the co-author and co-purchase networks due to GPU memory limit; hidden $\in \{64, 128, 256\}$; $L_2 \in \{5e-3, 1e-3, 8e-4, 5e-4, 1e-4, 5e-5, 1e-5\}$; dropout $\in \{0.1, 0.3, 0.5, 0.8\}$; $\alpha \in \{0.1, 0.2, 0.5\}$. As for the sampling process, $p_{\min}$ and $p'_{\max}$ are both selected from $\{0.05, 0.1, 0.2, \cdots, 1.0\}$.

B. Kernels

We list the detailed mathematical forms of the kernels as follows. The linear kernel $K_{\mathrm{Linear}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x}_2$; the polynomial kernel $K_{\mathrm{Poly}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x}_2)^2$; the radial basis (RBF) kernel $K_{\mathrm{RBF}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(l||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2)$ where we
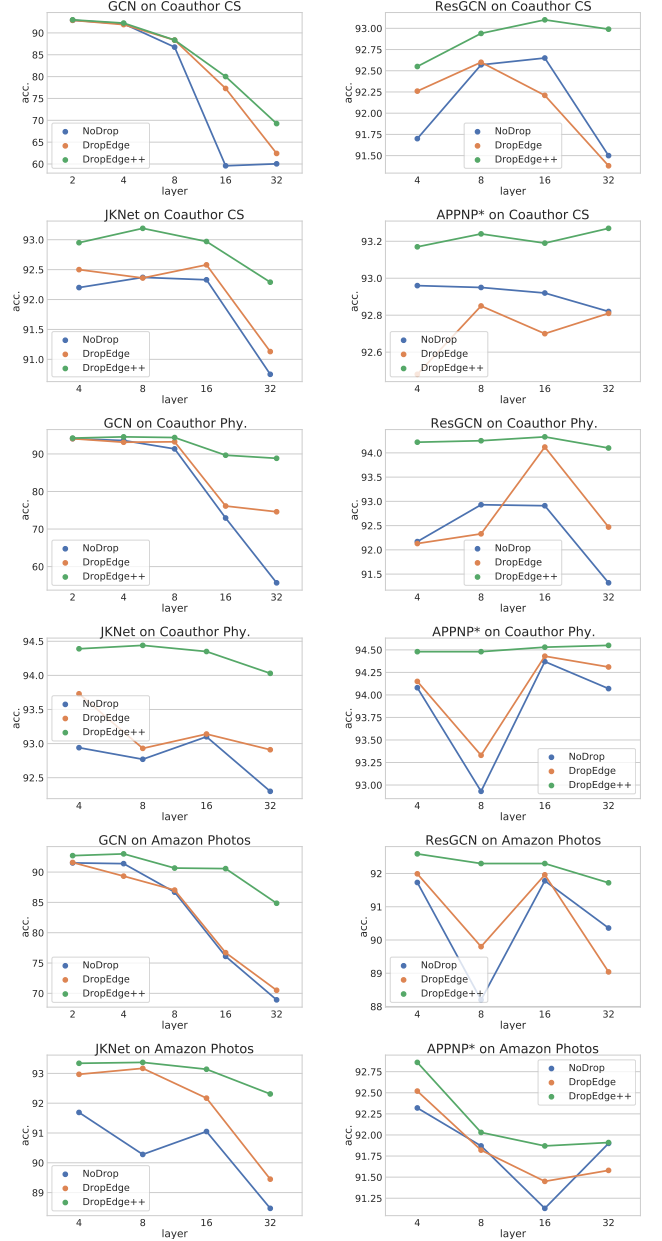

Fig. 6. Testing accuracy of varying models on Coauthor CS, Coauthor Phy, and Amazon Photos.

select $l = -6$ by validation. In our experiments, we use the Linear kernel by default.

TABLE X
HYPER-PARAMETERS FOR ACHIEVING THE BEST TESTING ACCURACY ON ALL DATASETS (EXCEPT APPNP*).

		Layer	hidden	lr	L_2	$p_{\min}$	$p'_{\max}$	norm.	drop.	BN	Loop
Cora (Full)	GCN	2	256	0.004	5e-4	0.1	0.4	ARW	0.3		
	ResGCN	4	256	0.004	5e-3	0.1	0.3	ARW	0.1		✓
	JKNet	8	256	0.006	8e-4	0.5	0.6	ARW	0.8		✓
Citeseer (Full)	GCN	2	256	0.006	1e-3	0.4	0.9	AN	0.5		✓
	ResGCN	16	256	0.010	5e-5	0.5	0.7	ANS	0.3		✓
	JKNet	4	128	0.009	5e-4	0.2	0.4	ANS	0.8		✓
Pubmed (Full)	GCN	2	256	0.001	1e-3	0.1	0.4	ANS	0.8	✓	✓
	ResGCN	8	256	0.003	1e-3	0.1	0.4	AN	0.8	✓	✓
	JKNet	8	128	0.005	1e-4	0.6	0.9	ARW	0.8	✓	✓
Cora (Semi)	GCN	4	256	0.009	1e-3	0.8	0.2	ANS	0.8		
	ResGCN	8	64	0.007	5e-5	0.6	1.0	ARW	0.8		
	JKNet	64	256	0.001	1e-3	0.05	1.0	AN	0.8		
Citeseer (Semi)	GCN	2	128	0.009	1e-3	0.8	0.8	ANS	0.8		
	ResGCN	64	64	0.007	1e-3	0.6	0.6	ARW	0.8		
	JKNet	8	128	0.007	1e-3	0.6	0.0	ARW	0.8		
Pubmed (Semi)	GCN	4	256	0.005	1e-3	0.4	0.5	NA	0.8		
	ResGCN	8	128	0.007	1e-3	0.6	1.0	ARW	0.8		
	JKNet	32	64	0.007	1e-3	0.6	0.8	ARW	0.8		
Coauthor CS	GCN	2	128	0.006	1e-3	0.2	0.7	ARW	0.5		✓
	ResGCN	16	128	0.009	5e-4	0.05	0.2	ARW	0.1		✓
	JKNet	8	128	0.003	5e-5	0.8	0.2	ANS	0.5		✓
Coauthor Physics	GCN	4	256	0.002	5e-5	0.3	0.3	AN	0.1		✓
	ResGCN	16	256	0.006	1e-5	0.05	1.0	ANS	0.3		
	JKNet	8	256	0.008	1e-5	0.7	0.8	ANS	0.8		
Amazon Photos	GCN	4	128	0.006	1e-5	0.2	0.1	FOG	0.5		✓
	ResGCN	4	256	0.006	1e-4	0.3	0.8	FOG	0.8		✓
	JKNet	8	256	0.010	1e-5	0.4	0.3	ARW	0.5		✓

TABLE XI
HYPER-PARAMETERS FOR ACHIEVING THE BEST TESTING ACCURACY ON ALL DATASETS WITH APPNP*.

	Layer	hidden	lr	L_2	$p_{\min}$	$p'_{\max}$	norm.	drop.	α
Cora (Full)	32	64	0.004	5e-5	0.5	1.0	ARW	0.5	0.2
Citeseer (Full)	16	128	0.010	5e-6	0.6	0.0	ARW	0.5	0.5
Pubmed (Full)	8	128	0.008	5e-5	0.9	0.6	ARW	0.3	0.5
Cora (Semi)	8	64	0.010	8e-4	0.9	0.8	NA	0.5	0.1
Citeseer (Semi)	32	64	0.010	8e-4	0.9	0.4	NA	0.5	0.1
Pubmed (Semi)	64	256	0.004	1e-4	0.3	0.2	ANS	0.1	0.5
Coauthor CS	8	64	0.008	5e-5	0.4	0.2	AN	0.5	0.2
Coauthor Physics	16	64	0.005	8e-6	0.7	0.5	AN	0.5	0.2
Amazon Photos	4	64	0.010	1e-5	0.7	0.5	ARW	0.1	0.2

APPENDIX E
MORE RESULTS

A. The co-author and co-purchase datasets

Fig. 6 displays the comparison on test accuracy with varying layers on Coauthor CS, Coauthor Physics, and Amazon Photos. In accordance with our analysis, DropEdge++ consistently boosts the performance of the backbones, and is very effective in formulating deep networks.

B. Mean-Edge-Number

Fig. 7 displays the metric, edge-number ($|\prod_{i=0}^{l} A_{\mathrm{drop}}^{(i)}|$), as the vertical axis and the number of layers as the horizontal axis on various layers on Cora and Citeseer, respectively, with $p_{\min} = 0.05, p_{\max} = 1.0, p = (p_{\min} + p_{\max})/2$. Colored area indicates the standard deviation over 10 runs. Table XII shows the values of MEN on Cora and Citeseer with different number of layers and sampling methods. Clearly, our LID sampler is very effective in reducing the MEN on all numbers of layers stacked, and is beneficial to relieving over-smoothing consequently.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
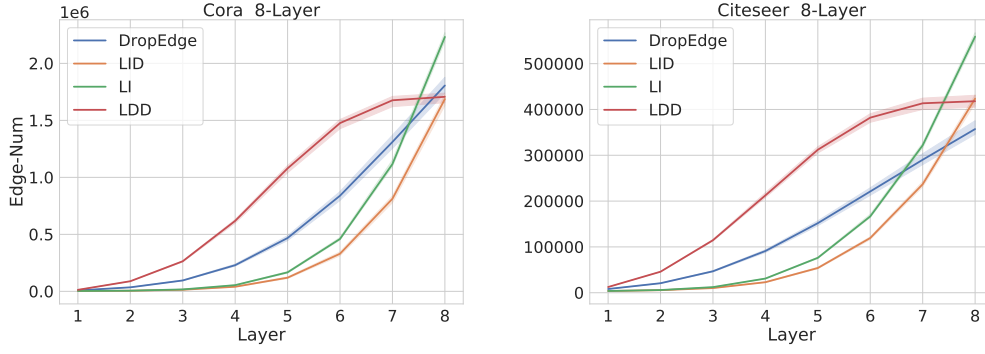
Fig. 7. The edge-number ($|\prod_{i=0}^{l} \boldsymbol{A}_{\mathrm{drop}}^{(i)}|$) comparison on Cora and Citeseer.

TABLE XII
THE VALUE OF MEN ON CORA AND CITESEER WITH VARYING LAYERS AND SAMPLING METHODS (10 RUNS).

	Layer	4	6	8	10	12	16
	DropEdge	9.0×10^4	2.7×10^5	6.0×10^5	9.4×10^5	1.3×10^6	1.8×10^6
Cora	LID	$\mathbf{5.7 \times 10^4}$	$\mathbf{1.7 \times 10^5}$	$\mathbf{3.8 \times 10^5}$	$\mathbf{6.3 \times 10^5}$	$\mathbf{8.8 \times 10^5}$	$\mathbf{1.3 \times 10^6}$
	LI	6.1×10^4	2.1×10^5	5.1×10^5	9.1×10^5	1.3×10^6	2.0×10^6
	LDD	1.1×10^5	3.6×10^5	8.7×10^5	1.5×10^6	2.2×10^6	3.2×10^6
	DropEdge	4.2×10^4	8.9×10^4	1.5×10^5	2.1×10^5	2.8×10^5	4.2×10^5
Citeseer	LID	$\mathbf{2.9 \times 10^4}$	$\mathbf{6.3 \times 10^4}$	$\mathbf{1.1 \times 10^5}$	$\mathbf{1.6 \times 10^5}$	$\mathbf{2.2 \times 10^5}$	$\mathbf{3.5 \times 10^5}$
	LI	3.1×10^4	7.7×10^4	1.5×10^5	2.4×10^5	3.7×10^5	6.8×10^5
	LDD	5.1×10^4	1.3×10^5	2.4×10^5	3.8×10^5	5.5×10^5	9.7×10^5

[2] F. Fouss, L. Yen, A. Pirotte, and M. Saerens, "An experimental investigation of graph kernels on a collaborative recommendation task," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 863–868.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *ICLR*, 2020.

[8] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[9] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," 2018.

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

[11] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," *arXiv preprint arXiv:1909.03211*, 2019.

[12] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.

[13] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *ICLR*, 2020.

[14] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, "Towards deeper graph neural networks with differentiable group normalization," *arXiv preprint arXiv:2006.06972*, 2020.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proceedings of International Conference on Learning Representations*, 2013.

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

[17] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[18] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2017.

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[21] A. Feng, C. You, S. Wang, and L. Tassiulas, "Kergnns: Interpretable graph neural networks with graph kernels," *arXiv preprint arXiv:2201.00491*, 2022.

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1025–1035.

[23] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[24] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[25] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1416–1424.

[26] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[27] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 4558–4567.

[28] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.

[29] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Tackling over-smoothing for general graph convolutional networks," *arXiv e-prints*, pp. arXiv–2008, 2020.

[30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.

[31] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[32] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[33] H. Chang, Y. Rong, T. Xu, W. Huang, S. Sojoudi, J. Huang, and W. Zhu, "Spectral graph attention network," 2020.

[34] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 6530–6539.

[35] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," 2019.

Tingyang Xu is a Senior researcher of Machine Learning Center in Tencent AI Lab. He obtained the Ph.D. degree from The University of Connecticut in 2017 and joined Tencent AI Lab in July 2017. In Tencent AI Lab, he is working on deep graph learning, graph generations and applying the deep graph learning model to various applications, such as molecular generation and rumor detection. His main research interests include social network analysis, graph neural networks, and graph generations, with particular focus on design deep and complex graph learning models for molecular generations. He has published several papers on data mining, machine learning top conferences KDD, WWW, NeurIPS, ICLR, CVPR, ICML, etc.

Jiaqi Han is currently a graduate student at Department of Computer Science and Technology, Tsinghua University. He received his undergraduate degree of computer science from Tsinghua University in 2021. His research interest primarily lies in leveraging graph neural networks for system modeling, including social networks as well as the intersection with complex physical and chemical systems. He has several publications on top-tier venues including ICLR, KDD, etc.

Fuchun Sun , IEEE Fellow, received the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 1997. He is currently a Professor with the Department of Computer Science and Technology and President of Academic Committee of the Department, Tsinghua University, deputy director of State Key Lab. of Intelligent Technology and Systems, Beijing, China. His research interests include intelligent control and robotics, information sensing and processing in artificial cognitive systems, and networked control systems. He was recognized as a Distinguished Young Scholar in 2006 by the Natural Science Foundation of China. He became a member of the Technical Committee on Intelligent Control of the IEEE Control System Society in 2006. He serves as Editor-in-Chief of International Journal on *Cognitive Computation and Systems*, and an Associate Editor for a series of international journals including the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS.

Wenbing Huang is now an assistant professor at Institute for AI Industry Research (AIR), Tsinghua University. He received his Ph.D. degree of computer science and technology from Tsinghua University in 2017. His current research mainly lies in the areas of machine learning, computer vision, and robotics, with particular focus on learning on irregular structures including graphs and videos. He has published about 30 peer-reviewed top-tier conference and journal papers, including the Proceedings of NeurIPS, ICLR, ICML, CVPR, etc. He served (will serve) as a Senior Program Committee of AAAI 2021, Area Chair of ACMMMM workshop HUMA 2020, and Session Chair of IJCAI 2019.

Junzhou Huang is an Associate Professor in the Computer Science and Engineering department at the University of Texas at Arlington. He also served as the director of machine learning center in Tencent AI Lab. He received the B.E. degree from Huazhong University of Science and Technology, Wuhan, China, the M.S. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree in Computer Science at Rutgers, The State University of New Jersey. His major research interests include machine learning, computer vision and imaging informatics. He was selected as one of the 10 emerging leaders in multimedia and signal processing by the IBM T.J. Watson Research Center in 2010. His work won the MICCAI Young Scientist Award 2010, the FIMH Best Paper Award 2011, the MICCAI Young Scientist Award Finalist 2011, the STMI Best Paper Award 2012, the NIPS Best Reviewer Award 2013, the MICCAI Best Student Paper Award Finalist 2014 and the MICCAI Best Student Paper Award 2015. He received the NSF CAREER Award in 2016.

Yu Rong is a Senior researcher of Machine Learning Center in Tencent AI Lab. He received the Ph.D. degree from The Chinese University of Hong Kong in 2016. He joined Tencent AI Lab in June 2017. His main research interests include social network analysis, graph neural networks, and large-scale graph systems. In Tencent AI Lab, he is working on building the large-scale graph learning framework and applying the deep graph learning model to various applications, such as ADMET prediction and malicious detection. He has published several papers on data mining, machine learning top conferences, including the Proceedings of KDD, WWW, NeurIPS, ICLR, CVPR, ICCV, etc.