EPIC: COMPRESSING DEEP GNNS VIA EXPRESSIVE POWER GAP-INDUCED KNOWLEDGE DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The teacher-student paradigm-based knowledge distillation (KD) has recently emerged as a promising technique for compressing graph neural networks (GNNs). Despite the great success in compressing moderate-sized GNNs, distilling deep GNNs (e.g., with over 100 layers) remains a tough challenge. A widely recognized reason is the *teacher-student expressive power gap*, i.e., the embeddings of a deep teacher may be extremely hard for a shallow student to approximate. Besides, the theoretical analysis and measurement of this gap are currently missing, resulting in a difficult trade-off between the needs of being "lightweight" and being "expressive" when selecting a student for the deep teacher. To bridge the theoretical gap and address the challenge of distilling deep GNNs, we propose the *first* GNN KD framework that quantitatively analyzes the teacher-student expressive power gap, namely **E**xpressive **P**ower gap-**I**ndu**C**ed knowledge distillation (**EPIC**). Our key idea is to formulate the estimation of the expressive power gap as an embedding regression problem based on the theory of polynomial approximation. Then, we show that the minimum approximation error has an upper bound, which decreases rapidly with respect to the number of student layers. Furthermore, we empirically demonstrate that the upper bound exponentially converges to zero as the number of student layers increases. Moreover, we propose to narrow the range of tuning the number of student layers based on the upper bound, and use an expressive power gap-related loss term to further encourage the student to generate embeddings similar to those of the teacher. Experiments on large-scale benchmarks demonstrate that EPIC can effectively reduce the numbers of layers of deep GNNs, while achieving comparable or superior performance. Specifically, for the 1,001-layer RevGNN-Deep, we reduce the number of layers by 94% and accelerate inference by roughly eight times, while achieving comparable performance in terms of ROC-AUC on the large-scale benchmark `ogbn-proteins`.

1 INTRODUCTION

Graph neural networks (GNNs) have recently achieved great success in many real-world applications involving graph data, such as electronic design automation (Ghose et al., 2021; Wang et al., 2022b), combinatorial optimization (Cappart et al., 2023; Peng et al., 2021), search engines (Zhu et al., 2021), recommendation systems (Fan et al., 2019; Wu et al., 2022c), and molecular property prediction (Wieder et al., 2020; Wang et al., 2022a). The key idea of GNNs is to iteratively update node embeddings based on both node features and graph structures (Gilmer et al., 2017; Hamilton, 2020; Kipf & Welling, 2017). Specifically, at each layer, GNNs aggregate messages from each node's neighborhood and then update node embeddings based on aggregation results and node features. Thus, the final embedding of a node in an L-layer GNN contains the information about its L-hop neighborhood (Hamilton, 2020).

Recently, deep GNNs (Li et al., 2021; Liu et al., 2020; Chen et al., 2020; Li et al., 2019; 2020) have shown leading performance on large-scale public benchmarks such as Microsoft Academic Graph (MAG) (Wang et al., 2020) and Open Graph Benchmark (OGB) (Hu et al., 2020), due to their powerful ability to learn long-range interactions (Chen et al., 2022; Cong et al., 2021). Many techniques improve the prediction performance of deep GNNs from aspects of graph convolutions (Li et al., 2021; Chen et al., 2020), normalization (Zhao & Akoglu, 2019; Guo et al., 2023a; Zhou et al., 2020; 2021), and initialization (JAISWAL et al., 2022) to address the challenges of oversmoothing (Li et al., 2019; Zeng et al., 2021), over-squashing (Topping et al., 2022), and information bottleneck (Alon & Yahav, 2021).

Nevertheless, the large numbers of layers in deep GNNs severely slow down their inference, making it difficult to deploy deep GNNs in latency-limited scenarios (Chen et al., 2021; Lee et al., 2019; Gong et al., 2021). Unlike models without graph dependency (e.g., Multi-layer perceptrons/Transformers (Vaswani et al., 2017)/CNNs (Krizhevsky et al., 2012)), deep GNNs suffer from the notorious neighbor explosion issue incurred by data dependency (Hamilton et al., 2017), i.e., the exponentially increasing dependencies of nodes with the number of layers. Therefore, knowledge distillation—a promising class of techniques for accelerating inference, while maintaining prediction performance—have recently become popular in real applications of GNNs.

Various knowledge distillation techniques for GNNs (KD4GNN) have achieved great success in compressing moderate-sized or shallow GNNs (e.g., GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and APPNP (Gasteiger et al., 2018). Knowledge distillation aims to train a lightweight “student” model under the supervision of a well-trained “teacher” model, where the supervisory signal (i.e., knowledge) can be anything computed by the teacher model (Gou et al., 2021). Most existing KD4GNN methods (Yang et al., 2022; Guo et al., 2023b; Yang et al., 2020b; Wu et al., 2022a; Zhang et al., 2020b; He et al., 2022) aim to distill GNNs into GNNs with fewer layers or parameters by improving the training framework, the selection of knowledge, or the configurations of teacher models. Another line of impressive works (Yang et al., 2021; Zhang et al., 2021; Tian et al., 2022) propose to distill GNNs into multi-layer perceptrons (MLPs) to avoid the dependency on graph structural information, e.g., neighbor nodes and edge features. The aforementioned works take the significant first step in exploring KD4GNN, and thus have drawn increasing attention in recent years.

However, distilling deep GNNs (e.g., with over 100 layers) remains a tough challenge. A widely recognized reason is that knowledge distillation may be ineffective when the *expressive power gap* between the teacher and student is large, as the embeddings of the teacher may be extremely hard for the student to approximate (Mirzadeh et al., 2020; Gao et al., 2021; Li & Leskovec, 2022). A typical example is distilling GNNs to MLPs. Because MLPs cannot distinguish two nodes with the same features and different neighbors, their expressive power is much weaker than GNNs, hence they are not “good students” for distilling deep GNNs. Besides, the theoretical analysis and measurement of the expressive power gap are currently missing, so it is difficult for us to select for the deep teacher a student that is shallow, while expressive.

In this paper, we aim to bridge the theoretical gap and address the challenge of distilling deep GNNs that *excel on large-scale graphs* and *possess numerous layers*. Towards this goal, we propose the *first* KD4GNN framework that quantitatively analyzes the teacher-student expressive power gap, namely **E**xpressive **P**ower gap-**I**ndu**C**ed Knowledge Distillation (**EPIC**). Our key idea is to formulate the estimation of the expressive power gap as an embedding regression problem based on the theory of polynomial approximation. Then, we show that the minimum approximation error has an upper bound, i.e., the EPIC bound, which decreases rapidly with respect to the number of student layers. Furthermore, our numerical experiments demonstrate that the EPIC bound exponentially converges to zero as the number of student layers increases, which empirically guarantees that we can distill deep GNNs into shallow GNNs. Moreover, we propose to narrow the range of tuning the number of student layers based on the EPIC bound, and use an expressive power gap-related loss to further encourage the student to generate embeddings similar to those of the teacher. Experiments on large-scale benchmarks demonstrate that EPIC can effectively reduce the numbers of layers of deep GNNs, while achieving comparable or superior performance. Specifically, for the 1,001-layer RevGNN-Deep, we reduce the number of layers by 94% and accelerate inference by roughly eight times, while achieving comparable performance in terms of ROC-AUC on the large-scale benchmark `ogbn-proteins`.

2 RELATED WORK

In this section, we discuss some works related to our proposed framework.

2.1 DEEP GRAPH NEURAL NETWORKS

Deep GNNs have strong potential to complete tasks involving large-scale graph-structured data due to their powerful ability to learn long-range interactions (Chen et al., 2022; Cong et al., 2021). Many works propose various approaches, including skip connection (Li et al., 2019; 2020; Xu et al., 2018;

Gasteiger et al., 2018; Chen et al., 2020), graph normalization (Ioffe & Szegedy, 2015; Zhao & Akoglu, 2019; Zhou et al., 2021; Yang et al., 2020a; Zhou et al., 2020), random dropping (Rong et al., 2019; Huang et al., 2020), and grouped reversible graph connections (Li et al., 2021), to train powerful deep GNNs on large-scale graphs (Chen et al., 2022). Various deep GNNs (Li et al., 2021; Liu et al., 2020; Chen et al., 2020; Li et al., 2019; 2020) have shown leading performance on large-scale public benchmarks such as Microsoft Academic Graph (MAG) (Wang et al., 2020) and Open Graph Benchmark (OGB) (Hu et al., 2020). It is worth mentioning that the 1,001-layer RevGNN-Deep trained on `ogbn-proteins` is the deepest GNN that has been published so far. Despite the great success on public benchmarks, deep GNNs suffer from the notorious neighbor explosion problem incurred by the data dependency (Hamilton et al., 2017), i.e., the exponentially increasing dependencies of nodes with the number of layers. This poses significant challenges to employing deep GNNs in latency-limited scenarios. Therefore, we aim to compress deep GNNs via the promising teacher-student paradigm-based knowledge distillation (KD) techniques.

2.2 KNOWLEDGE DISTILLATION FOR GNNS (KD4GNN)

To compress GNNs, many works propose various KD (Hinton et al., 2015) techniques, which aims to train lightweight “student” models under the supervision of well-trained “teacher” models.

GNNs-to-GNNs Distillation. Most existing KD4GNN works aim to distill GNNs to GNNs with fewer layers or parameters. According to their key ideas for improving the distillation, we can classify them into three categories: (1) what to distill (how to select appropriate knowledge) (Yang et al., 2022; Huo et al., 2023; Yan et al., 2020; Yang et al., 2020b; Zhang et al., 2020b; He et al., 2022), (2) how to distill (how to improve the training paradigm to better transfer knowledge) (Guo et al., 2023b; Zhang et al., 2020b; He et al., 2022), and (3) who will teach (how to select appropriate teachers) (Huo et al., 2023; Guo et al., 2023b; Zhang et al., 2020b).

GNNs-to-MLPs Distillation. Another line of impressive works (Yang et al., 2021; Zhang et al., 2021; Tian et al., 2022; Wu et al., 2023) propose to distill GNNs into multi-layer perceptrons (MLPs) to avoid the dependency on graph structural information, such as neighbor nodes and edge features. Because MLPs take only node features as input, their inference speed is much faster than GNNs. However, as shown in Section 6.2, the distillation performance of these methods in the inductive setting is unsatisfactory, as the expressive power of MLPs is much weaker than GNNs and MLPs do not have access to the teachers’ soft labels on test nodes to use as guidance.

Distillation for Deep GNNs. The aforementioned works have achieved great success in distilling moderate-sized or shallow GNNs (e.g., GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and APPNP (Gasteiger et al., 2018). However, distilling deep GNNs (e.g., with over 100 layers) remains a tough challenge. A widely recognized reason is that KD may be ineffective when the expressive power gap between the teacher and student is large, as the embeddings of the teacher may be extremely hard for the student to approximate (Mirzadeh et al., 2020; Gao et al., 2021; Li & Leskovec, 2022). Besides, the theoretical analysis and measurement of this gap are currently missing. Thus, we aim to bridge the theoretical gap and address the challenge of distilling deep GNNs that *excel on large-scale graphs* and *possess numerous layers*. To the best of our knowledge, this is the *first* paper that attempts to distill GNNs with more than 100 layers.

2.3 EXPRESSIVE POWER OF GRAPH NEURAL NETWORKS

The expressive power of GNNs has attracted much attention recently. For example, (Xu et al., 2019) and (Morris et al., 2019) show that message passing-based GNNs are at most as powerful as 1-WL test (Weisfeiler & Leman, 1968) to distinguish non-isomorphic graphs. Thus, many works propose various techniques to increase the power of GNNs from the perspective of higher-order WL tests (Morris et al., 2019; Maron et al., 2019; Chen et al., 2019) and graph bi-connectivity (Zhang et al., 2023). However, the above-mentioned works mainly focus on GNNs’ whole-graph expressive power. Therefore, to analyze the GNNs’ link expressive power, (Zhang et al., 2020a) proposes a multi-node representation theory and explains the superior performance of the labeling trick used in GNNs for link prediction. Besides, for node property prediction, (Wang & Zhang, 2022) shows that spectral GNNs can produce arbitrary graph signals under some mild conditions.

3 PRELIMINARIES

We introduce notations, GNNs, and KD in Sections 3.1, 3.2, and 3.3, respectively.

3.1 NOTATIONS

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by an unordered set of nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, where n is the number of nodes, and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ among these nodes. The node set $\mathcal{V} = \mathcal{V}^{\mathrm{L}} \sqcup \mathcal{V}^{\mathrm{U}}$ is the disjoint union of the labeled node set $\mathcal{V}^{\mathrm{L}}$ and the unlabeled node set $\mathcal{V}^{\mathrm{U}}$. The label of a node $v_i \in \mathcal{V}^{\mathrm{L}}$ is y_i. Let $(v_i, v_j) \in \mathcal{E}$ denote an edge going from $v_i \in \mathcal{V}$ to $v_j \in \mathcal{V}$ and $\mathcal{N}(v_i) = \{v_j \in \mathcal{V} | (v_i, v_j) \in \mathcal{E}\}$ denote the neighborhood of v_i. Let $\mathbf{A} \in \{0, 1\}^{n \times n}$ be the adjacency matrix of $\mathcal{G}$ ($\mathbf{A}_{i,j} = 1$ if and only if $(v_i, v_j) \in \mathcal{E}$, $i, j \in [n]$) and $\mathbf{D}$ be the diagonal matrix whose diagonal element $\mathbf{D}_{i,i}$ is the degree of v_i, $i \in [n]$. We assume that $\mathcal{G}$ is undirected, i.e., $v_j \in \mathcal{N}(v_i) \Leftrightarrow v_i \in \mathcal{N}(v_j)$, hence $\mathbf{A}$ is symmetric. Denote the normalized adjacency matrix by $\widehat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. Let $\mathbf{I}$ be the identity matrix, and denote the normalized Laplacian matrix by $\widehat{\mathbf{L}} = \mathbf{I} - \widehat{\mathbf{A}}$, whose eigendecomposition is $\widehat{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is the orthogonal matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. In some scenarios, nodes are associated with a node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d_x}$. For a positive integer L, $[L]$ denotes $\{1, \ldots, L\}$. For $i \in [n]$, let $\mathbf{X}_{i,:} \in \mathbb{R}^{d_x}$ and $\mathbf{H}_{i,:} \in \mathbb{R}^{d_h}$ denote the feature and the embedding of the node v_i with dimension d_x and d_h, respectively. Let $\mathbf{H} = \left(\mathbf{H}_{1,:}^\top, \ldots, \mathbf{H}_{n,:}^\top\right)^\top \in \mathbb{R}^{n \times d_h}$ be the embedding matrix of the graph. We also denote the embeddings of a set of nodes $\mathcal{S} = \{v_{i_k}\}_{k=1}^{|\mathcal{S}|}$ by $\mathbf{H}_{\mathcal{S}} = (\mathbf{H})_{\mathcal{S}} = \left(\mathbf{H}_{i_1,:}^\top, \ldots, \mathbf{H}_{i_k,:}^\top\right)^\top \in \mathbb{R}^{d_h \times |\mathcal{S}|}$. For a matrix $\mathbf{W} \in \mathbb{R}^{p \times q}$, we denote its i-th row by $\mathbf{W}_{i,:}$ and j-th column by $\mathbf{W}_{:,j}$, where $i \in [p]$ and $j \in [q]$, respectively.

3.2 GRAPH NEURAL NETWORKS

In this paper, we focus on node-level tasks on graphs, which aim to predict a discrete or continuous label for each node. Graph neural networks (GNNs) iteratively update node embeddings based on node features and graph structures (Hamilton, 2020; Kipf & Welling, 2017). At each layer, GNNs aggregate messages from each node's neighborhood and then update node embeddings based on aggregation results and node features.

A GNN with L layers and parameters $(\theta^{(l)})_{l=1}^{L}$ generates the final node embeddings $\mathbf{H} = \mathbf{H}^{(L)}$ as

$$\mathbf{H}^{(l)} = f_{\theta^{(l)}}(\mathbf{H}^{(l-1)}; \mathbf{X}, \mathbf{A}), \ l \in [L], \tag{1}$$

where $\mathbf{H}^{(0)} = \mathbf{X}$ and $f_{\theta^{(l)}}$ is the l-th layer with parameters $\theta^{(l)}$.

3.3 KNOWLEDGE DISTILLATION

Knowledge Distillation (KD) aims to train a lightweight student model S by transferring the knowledge from a well-trained teacher model T (Tian et al., 2023). The key idea of KD is to encourage S to mimic the behaviors of T under its supervision, where the supervisory signal (i.e., knowledge) can be anything computed by T, such as logits and hidden embeddings (Gou et al., 2021).

Specifically, given the knowledge k_T and k_S computed by T and S, respectively, we define the knowledge distillation loss as $\mathcal{L}_{kd} = \mathrm{dist}(k^T, k^S)$, where $\mathrm{dist}(\cdot, \cdot)$ is a distance function (note that we do not require it to be symmetric), such as the Kullback-Leibler divergence and the Euclidean distance. For example, for the vanilla knowledge distillation proposed in (Hinton et al., 2015), the knowledge is the soft labels derived by applying the Softmax operation on logits, and the distance function is the Kullback-Leibler divergence.

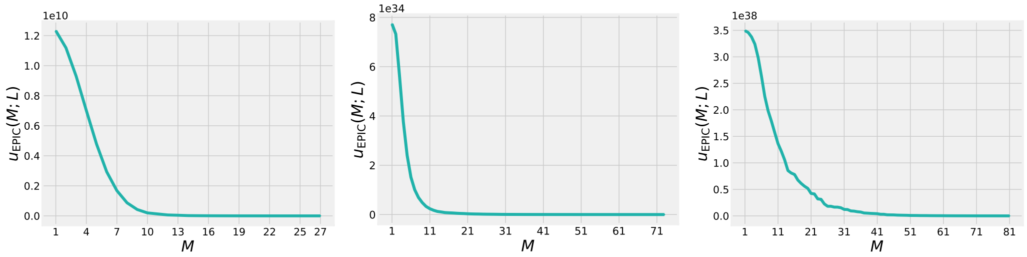
4 THEORETICAL ANALYSIS OF EXPRESSIVE POWER GAP

In the knowledge distillation of deep GNNs, a common but knotty problem is how to select an appropriate value for the number of student layers, given a deep teacher. This problem involves another one that is more essential, i.e., *how does the number of layers of one GNN affect its expressive power?* Specifically, deep L-layer teacher GNNs encode the information about nodes' L-hop neighbors (Hamilton, 2020), while shallow M-layer student GNNs are difficult to encode the long-range interactions as well as the deep teacher if $M \ll L$. To address this problem, we theoretically analyze the gap between the expressive power of GNNs with different numbers of layers in this section.

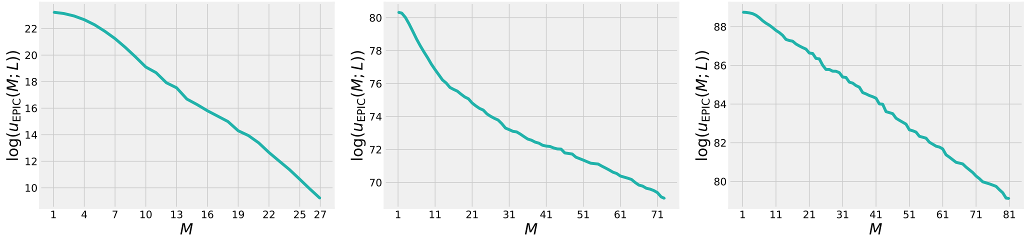
We analyze the expressive power of GNNs on node-level tasks from a spectral perspective following (Wang & Zhang, 2022). Given a graph $\mathcal{G}$ with node features $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ and the normalized Laplacian matrix $\widehat{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and $\mathbf{U} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, a spectral-based L-layer GNN generates node embeddings at the l-th layer as

$$\mathbf{H}^{(l)} = g_{\mathbf{a}}^{(l)}(\widehat{\mathbf{L}})\phi_{\mathbf{w}}(\mathbf{X}), \ l \in [L], \tag{2}$$

where $g_{\mathbf{a}}^{(l)}(t) = \sum_{k=0}^{l} a_k t^k$ is an l-degree polynomial with parameters $\mathbf{a} = (a_k)_{k=0}^{l}$ and $\phi_{\mathbf{w}}$ is a neural network with parameters $\mathbf{w}$. In the theoretical analysis, we suppose that the following assumptions



(a) u_{EPIC} on ogbn-arxiv (b) u_{EPIC} on ogbn-products (c) u_{EPIC} on ogbn-proteins



(d) $\log(u_{\mathrm{EPIC}})$ on ogbn-arxiv (e) $\log(u_{\mathrm{EPIC}})$ on ogbn-products (f) $\log(u_{\mathrm{EPIC}})$ on ogbn-proteins

Figure 1: $u_{\mathrm{EPIC}}(M;L)$ and $\log(u_{\mathrm{EPIC}}(M;L))$ with respect to M on three large-scale datasets.

hold in this paper. For the relationship of our spectral-based theoretical analysis to non-linear GNNs, please refer to Appendix B.1.

Assumption 1. *We assume that (1) the n eigenvalues $(\lambda_i)_{i=1}^{n}$ of $\widehat{\mathbf{L}}$ are different from each other, (2) there exists $C > 0$ such that $\|\phi_{\mathbf{w}}(\mathbf{X})\|_F < C/n^2$ for any $\mathbf{w}$.*

Given a well-trained L-layer GNN denoted $G_T^{(L)}$ with embeddings $\mathbf{H}_T^{(L)} \in \mathbb{R}^{n \times d_h}$ and an M-layer GNN denoted $G_S^{(M)}$ with embeddings $\mathbf{H}_S^{(M)} \in \mathbb{R}^{n \times d_h}$, where $M < L$, we next analyze their expressive power gap. Suppose that the embeddings are computed by

$$\mathbf{H}_T^{(L)} = g_{\mathbf{a}}^{(L)} \phi_{\mathbf{w}_T}(\mathbf{X}), \quad \mathbf{H}_S^{(M)} = g_{\mathbf{b}}^{(M)} \phi_{\mathbf{w}_S}(\mathbf{X}), \tag{3}$$

respectively. We formulate the estimation of their expressive power gap as finding the minimum approximation error of $\mathbf{H}_S^{(M)}$ to $\mathbf{H}_T^{(L)}$ in terms of the Frobenius norm, i.e.,

$$e(M;L) \triangleq \min_{\mathbf{b},\mathbf{w}_S} \|\mathbf{H}_S^{(M)} - \mathbf{H}_T^{(L)}\|_F, \tag{4}$$

The following theorem gives an upper bound of $e(M;L)$ and shows that the upper bound decreases monotonically with respect to M. For the detailed proof, please refer to Appendix B.

Theorem 1. *Given an n-node graph $\mathcal{G}$ with node features $\mathbf{X}$ and the normalized Laplacian matrix $\widehat{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, an L-layer well-trained GNN ($G_T^{(L)}$) and an M-layer GNN ($G_S^{(M)}$) that compute embeddings as shown in Eq. (3), we suppose that Assumption 1 holds, then we have*

$$e(M;L) \leq u_{\mathrm{EPIC}}(M;L) \triangleq C\|\mathbf{P}^{(M)}(\mathbf{P}^{(M)})^\top \mathbf{d}^{(M)} - \mathbf{d}^{(M)}\|_2, \tag{5}$$

*where $e(M;L)$ is defined by Eq. (4), $u_{\mathrm{EPIC}}(M;L)$ is named the **EPIC bound**, and $\mathbf{P}^{(M)} \in \mathbb{R}^{n \times (M+1)}$ is the left-singular matrix of*

$$\mathbf{V}^{(M)} = \begin{pmatrix} 1 & \lambda_1 & \cdots & \lambda_1^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_n & \cdots & \lambda_n^M \end{pmatrix}, \tag{6}$$

and

$$\mathbf{d}^{(M)} = \left(\sum_{k=M+1}^{L} a_k \lambda_1^k, \ldots, \sum_{k=M+1}^{L} a_k \lambda_n^k \right)^\top. \tag{7}$$

Moreover, $u_{\mathrm{EPIC}}(M;L)$ decreases monotonically with respect to M. Specifically, we have $u_{\mathrm{EPIC}}(L;L) = 0$, which leads to $e(L;L) = 0$.

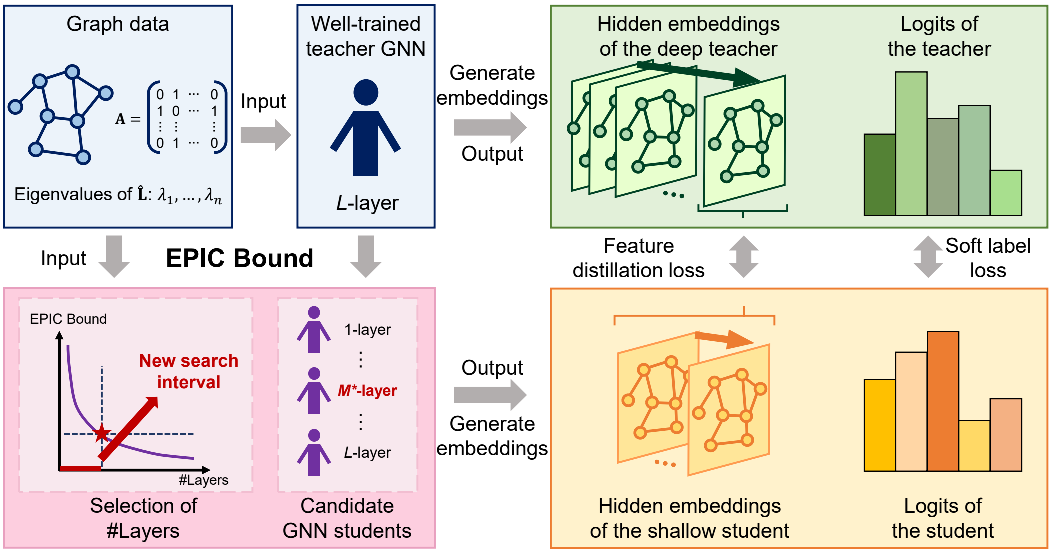


Figure 2: The overall framework of EPIC. Given a graph $\mathcal{G}$ and a well-trained L-layer teacher GNN, we compute EPIC bounds for different values of numbers of student layers and narrow the range of tuning the number of student layers (see Section 5.1). Then, we use an expressive power gap-related loss (i.e., a feature distillation loss) to further encourage the student to generate embeddings similar to those of the teacher (see Section 5.2).

Please note that the monotonically decreasing property of $u_{\text{EPIC}}(M; L)$ does not depend on L and $\mathbf{a} = (a_k)_{k=0}^{L}$. To further study how the EPIC bound $u_{\text{EPIC}}(M; L)$ decreases as M increases, we conduct numerical experiments on large-scale datasets `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`. As shown in Figure 1, $u_{\text{EPIC}}(M; L)$ converges exponentially to zero (note that $u_{\text{EPIC}}(L; L) = 0$) as M increases. This empirically guarantees that we can distill deep GNNs into shallow GNNs. For more details about the experiments, please refer to Section 6.4.

5 EXPRESSIVE POWER GAP-INDUCED KNOWLEDGE DISTILLATION

Based on our theoretical analysis in Section 4, we propose a GNN KD framework that takes the teacher-student expressive power gap into account, namely Expressive Power Gap-Induced Knowledge Distillation (EPIC). Figure 2 illustrates the overall framework of EPIC. Specifically, given a graph $\mathcal{G}$ and a well-trained deep GNN with L layers, we first compute EPIC bounds for different values of numbers of student layers and select an appropriate value for the number of student layers such that the student is shallow, while expressive. Then, to further encourage the student to generate embeddings similar to those of the teacher, we propose an expressive power gap-induced loss term.

5.1 SELECTION OF NUMBER OF STUDENT LAYERS

Pre-processing: Computing EPIC Bounds. Given an L-layer teacher GNN ($G_T^{(L)}$) and an n-node training graph $\mathcal{G}$ with the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, we first compute the normalized Laplacian $\widehat{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{n \times n}$, where $\mathbf{D}$ is the degree matrix, and the eigenvalues $(\lambda_i)_{i=1}^{n}$ of $\widehat{\mathbf{L}}$.

Then, for each $M \in [L-1]$, we compute the matrix $\mathbf{V}^{(M)} \in \mathbb{R}^{n \times (M+1)}$ defined by Eq. (6) and its singular value decomposition (SVD) $\mathbf{V}^{(M)} = \mathbf{P}^{(M)}\mathbf{\Sigma}^{(M)}(\mathbf{Q}^{(M)})^{\top}$, where $\mathbf{P}^{(M)} \in \mathbb{R}^{n \times (M+1)}$, $\mathbf{\Sigma}^{(M)} \in \mathbb{R}^{(M+1) \times (M+1)}$, and $\mathbf{Q}^{(M)} \in \mathbb{R}^{(M+1) \times n}$. If $G_T^{(L)}$ is a spectral GNN, we use its real parameters as $\mathbf{a} = (a_k)_{k=0}^{L}$. Otherwise, we let $a_k = 1$ for $k = 0, \dots, L$. Note that this does not affect the monotonically decreasing property of the EPIC bound. Then, we compute the vector $\mathbf{d}$ defined by Eq. (7) and the EPIC bound $u_{\text{EPIC}}(M; L)$ by Eq. 5 (we can simply let $C = 1$).

It is worth noting that we do not need to compute EPIC bounds in the inference stage of GNNs, hence the computation of EPIC bounds do not affect the inference speed of GNNs, which is the focus of this paper and most knowledge distillation methods.

Narrowing the Range of Tuning the Number of Student Layers. After computing EPIC bounds $(u_{\text{EPIC}}(M; L))_{M=1}^{L-1}$, we plot $u_{\text{EPIC}}(M; L)$ as a function of M. By observing the plot, we find the maximum value $M_{\max}$ such that $u_{\text{EPIC}}(M; L)$ decreases slowly when $M > M_{\max}$. Then we tune the number M of student layers in $[1, M_{\max}]$, which is much smaller than $[1, L]$ since $u_{\text{EPIC}}(M; L)$ converges exponentially to zero as M increases.

5.2 EXPRESSIVE POWER GAP-RELATED LOSS

To further improve the expressive power of the student $G_S^{(M)}$, we concatenate its embeddings at top-K layers and multiply it by a linear mapping matrix $\mathbf{W} \in \mathbb{R}^{Kd_S \times d_T}$ as

$$\widehat{\mathbf{H}}_S^{(M)} = \left[\mathbf{H}_S^{(M-K+1)} \quad \cdots \quad \mathbf{H}_S^{(M)}\right] \mathbf{W} \in \mathbb{R}^{n \times Kd_T}, \tag{8}$$

where K is a hyperparameter, d_S is the width of $G_S^{(M)}$, and d_T is the width of the teacher $G_T^{(L)}$. We take $\widehat{\mathbf{H}}_S^{(M)}$ as the final embedding matrix of $G_S^{(M)}$.

To encourage $G_S^{(M^*)}$ to generate embeddings similar to those of $G_T^{(L)}$, we use an expressive power gap-related loss (feature distillation loss), i.e.,

$$\mathcal{L}_{\text{EP}} = \|(\widehat{\mathbf{H}}_S^{(M)})_{\mathcal{V}^{\text{tr}}} - (\mathbf{H}_T^{(L)})_{\mathcal{V}^{\text{tr}}}\|_F^2, \tag{9}$$

where $\mathcal{V}^{\text{tr}}$ is the train set. Besides, we also use the ground truth loss and the soft label loss, i.e.,

$$\mathcal{L}_{\text{GT}} = \sum_{v \in \mathcal{V}^{\text{tr}}} \text{CE}(\widehat{\mathbf{y}}_v, \mathbf{y}_v), \quad \mathcal{L}_{\text{SL}} = \sum_{v \in \mathcal{V}^{\text{tr}}} D_{\text{KL}}(\widehat{\mathbf{y}}_v, \mathbf{z}_v), \tag{10}$$

where $\text{CE}(\cdot, \cdot)$ is the cross-entropy function, $D_{\text{KL}}(\cdot, \cdot)$ is the Kullback-Leibler divergence, $\widehat{\mathbf{y}}$ is the prediction of $G_S^{(M)}$, $\mathbf{y}$ is the vector of ground truth labels, and $\mathbf{z}$ is the vector of soft labels predicted by $G_T^{(L)}$. The final loss function $\mathcal{L}$ is the weighted sum of the three loss terms, i.e.,

$$\mathcal{L} = \mathcal{L}_{\text{GT}} + \lambda \mathcal{L}_{\text{SL}} + \mu \mathcal{L}_{\text{EP}}, \tag{11}$$

where λ and μ are the weight coefficients of loss terms. We summarize EPIC in Algorithm 1. Please note that it does not involve the process of updating parameters.

6 EXPERIMENTS

We first introduce experimental settings in Section 6.1. We then show the main results of EPIC on distilling deep GNNs on large-scale benchmarks in Section 6.2. After that, in Section 6.3, we conduct experiments to study the decreasing trend of the EPIC bound. Finally, we provide ablation studies in Section 6.4. We run all experiments on a single GeForce RTX 3090 Ti (24 GB).

Algorithm 1 EPIC: Expressive Power Gap-Induced Knowledge Distillation

```
Input: The training graph $\mathcal{G}$ with nodes $\mathcal{V}^{\text{tr}}$, node features $\mathbf{X}$, edge features $\mathbf{E}$ (if $\mathcal{G}$ has them), and the adjacency matrix $\mathbf{A}$. The ground truth labels $(\mathbf{y})_{\mathcal{V}^{\text{tr}}}$, soft labels $(\mathbf{z})_{\mathcal{V}^{\text{tr}}}$ and embeddings $(\mathbf{H}_T^{(L)})_{\mathcal{V}^{\text{tr}}}$ of the teacher $G_T^{(L)}$. The well-trained parameters $\mathbf{a} = (a_k)_{k=0}^L$ (if $G_T^{(L)}$ is spectral). The weight coefficients $\lambda$ and $\mu$. The number $K$ of concatenated embeddings and ratio $\gamma$.
Compute $\widehat{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and eigenvalues $(\lambda_i)_{i=1}^n$
for $M = 1, \ldots, L-1$ do
    Compute $\mathbf{V}^{(M)}$ by Eq. (6)
    Compute the SVD by $\mathbf{V}^{(M)} = \mathbf{P}^{(M)}\mathbf{\Sigma}^{(M)}(\mathbf{Q}^{(M)})^\top$
    Compute $\mathbf{d}^{(M)}$ by Eq. (7)
    Compute $u_{\text{EPIC}}(M; L)$ by Eq. (5)
end for
Find $M_{\max}$ by observing the plot of $u_{\text{EPIC}}(M; L)$
for $M = 1, \ldots, M_{\max}$ do
    Compute $\widehat{\mathbf{H}}_S^{(M)}$ by Eq. (8)
    Compute $\mathcal{L}_{\text{EP}}$, $\mathcal{L}_{\text{GT}}$, $\mathcal{L}_{\text{SL}}$, and $\mathcal{L}$ by Eqs. (9)-(11)
end for
```

6.1 EXPERIMENTAL SETTINGS

We conduct all experiments in the practical setting of the *inductive* training, i.e., test nodes are strictly unseen during training (Wu et al., 2022b). Specifically, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node features $\mathbf{X}$ and ground truth labels $\mathbf{y}$, where $\mathcal{V} = \mathcal{V}^{\text{L}} \sqcup \mathcal{V}^{\text{U}}$ is the disjoint union of the labeled node set $\mathcal{V}^{\text{L}}$ and the unlabeled node set $\mathcal{V}^{\text{U}}$, we remove all edges connected to nodes in $\mathcal{V}^{\text{U}}$ from $\mathcal{E}$ to form $\mathcal{E}^{\text{L}}$ and $\mathcal{G}^{\text{L}} = (\mathcal{V}^{\text{L}}, \mathcal{E}^{\text{L}})$. For details of the inductive setting, please refer to Appendix A.2.

For the *transductive* setting, instead of training a student model, we can store the final node embedding matrix $\mathbf{H}_T^{(L)}$ of teachers in cheap RAM or hard drive storage to accelerate inference, whose

Table 1: **Distillation performance of EPIC and baselines in the inductive setting on three large-scale datasets.** The "#Layers" refers to the number of GNN-layers. The "Perf." refers to the performance, whose metric is Accuracy for `ogbn-arxiv` and `ogbn-products`, and ROC-AUC for `ogbn-proteins`. The "#Layers↓" refers to the relative decrease of numbers of graph convolutional layers compared to the teacher.

Datasets	**Teacher**		**LSP-GCN**		**GLNN**	**NOSMOG**	**EPIC (Ours)**		
	#Layers	Perf.	#Layers	Perf.	Perf.	Perf.	#Layers	#Layers↓	Perf.
`Arxiv`	28	72.91±0.00	4	67.12±0.33	56.16±0.36	62.46±0.39	3	89.29%	**73.06±0.13**
`Products`	112	77.86±0.00	4	72.43±0.28	60.11±0.06	-	7	93.75%	**78.58±0.27**
`Proteins`	1,001	**85.91±0.08**	4	-	74.40±0.09	60.63±1.67	60	94.01%	85.85±0.09

runtime is marginal and space complexity $\mathcal{O}(|\mathcal{V}|)$ is similar to the node features $\mathbf{X}$. Therefore, the acceleration of inference without performance degradation under the transductive setting is trivial.

To distill deep GNNs on large-scale graphs and enforce a fair comparison, we select the datasets, the teacher models, the baselines, and the hyperparameters as follows.

Datasets. We evaluate the proposed EPIC on three large-scale OGB (Hu et al., 2020) datasets, i.e., `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`. These datasets have more than 100,000 nodes and 1,000,000 edges, and have become standard large-scale benchmarks in recent years. For more details, please refer to Appendix A.1.

Teacher Models. For `ogbn-products` and `ogbn-proteins`, we select the one with the highest ranking on the public leaderboard among GNNs with over 100 layers. For the relatively smaller `ogbn-arxiv`, we select the one with the highest ranking on the public leaderboard among GNNs with over 10 layers. Based on these criteria, we select RevGCN-Deep (with 28 layers) (Li et al., 2021), RevGNN-112 (with 112 layers) (Li et al., 2021), and RevGNN-Deep (with 1,001 layers) (Li et al., 2021) as our teacher models on `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, respectively. For more details, please refer to Appendix A.3.

Baselines. We compare EPIC with three representative KD4GNN frameworks, i.e., LSP (GNNs-to-GNNs) (Yang et al., 2020b), GLNN (GNNs-to-MLPs) (Zhang et al., 2021), and NOSMOG (GNNs-to-MLPs) (Tian et al., 2022). For more details, please refer to Appendix A.4 and A.5.

Hyperparameters. We follow the most of hyperparameters used to train our teachers (Li et al., 2021), except for the additional hyperparameters in EPIC such as the width of the student d_S, the weight coefficients λ and μ, the number K of concatenated embeddings, and the ratio γ for selecting the number of student layers. For a fair comparison, We use the grid search to find the best hyperparameters for EPIC and the three baselines (please see Appendix A.4 and A.6 for more details).

6.2 MAIN RESULTS

Distillation Performance of EPIC. In this section, we refer to the student(s) trained with EPIC as EPIC(s). Table 1 reports the distillation performance of EPIC and baselines in the inductive setting on three large-scale datasets. The numbers of the EPIC student layers are selected in the narrowed range based on EPIC bounds. We train each teacher for only once, as training deep GNNs takes a long time. For RevGNN-Deep on `ogbn-proteins`, we report the mean and standard deviation by running 50 mini-batch inferences with different random seeds. For the students, we train each one for five times with different seeds, run 50 mini-batch inferences with different seeds for each trained student (we run full-batch inference if the GPU memory can afford it), and report the mean and standard deviation.

As shown in Table 1, EPIC reduces the numbers of teacher layers by at least 89.29%, while achieving comparable or even superior performance. Specifically, on `ogbn-arxiv` and `ogbn-products`, EPICs outperform their teachers with relative decreases of numbers of layers by 89.29% and 93.75%, respectively. On `ogbn-proteins`, the performance of EPIC is slightly lower than its 1,001-layer teacher, while its relative decrease of the number of layers is as high as 94%.

To further demonstrate the effectiveness of EPIC, we compare it with LSP (Yang et al., 2020b), GLNN (Zhang et al., 2021), and NOSMOG (Tian et al., 2022). For LSP, we follow the original paper (Yang et al., 2020b) to use 4-layer GCNs as students. We do not report the result of LSP-GCN on `ogbn-proteins`, as the vanilla GCN is difficult to encode graphs with edge features. For GLNN and NOSMOG, the students are MLPs. We do not report the result of NOSMOG on `ogbn-products`, as it takes more than 50 hours to generate positional encodings for test nodes, which is much longer than the overall inference time of the teacher. As shown in Table 1, the performance of baselines in distilling deep GNNs is unsatisfactory, although they have achieved success in distilling moderate-sized GNNs. We attribute the reason to the weak expressive power of GCNs and MLPs. When distilling deep GNNs, the embeddings of teachers are extremely hard for them to approximate, hence the knowledge distillation becomes ineffective. Besides, in the inductive setting, shallow students do not have access to the teachers' soft labels on test nodes to use as guidance, which further poses challenges to knowledge distillation.

Table 2: **Comparisons of inference time between EPICs and deep teachers.** The unit of time in the table is seconds. The "Spd.↑" refers to the relative improvements of inference speed brought by EPIC.

Datasets	`Arxiv`	`Products`	`Proteins`
Teacher	$0.52_{\pm 0.01}$	$14.17_{\pm 0.05}$	$113.29_{\pm 0.75}$
EPIC	$0.08_{\pm 0.00}$	$1.06_{\pm 0.04}$	$14.11_{\pm 0.42}$
Spd.↑	6.50×	13.37×	8.03×

Table 3: **The absolute improvements of performance brought by $\mathcal{L}_{\mathrm{SL}}$ and $\mathcal{L}_{\mathrm{EP}}$.** The improvements are denoted Δ_{SL} and Δ_{EPIC}, respectively.

Datasets	`Arxiv`	`Products`	`Proteins`
Vanilla EPIC	72.91	77.71	85.00
$\mathcal{L}_{\mathrm{SL}}$	72.94	77.83	85.25
Δ_{SL}	0.03	0.12	0.25
$\mathcal{L}_{\mathrm{SL}} + \mathcal{L}_{\mathrm{EP}}$	73.06	78.58	85.85
Δ_{EPIC}	0.12	0.75	0.60

Acceleration Effect of EPIC. Table 2 shows the inference time of EPICs and their teachers. We report the mean and standard deviation by running inference for five times with different random seeds. For the deep teachers on `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, EPIC achieves speedups of 6.50×, 14.06× and 8.03×, respectively. This demonstrates the effectiveness of EPIC in accelerating the inference of deep GNNs.

6.3 ANALYSIS OF EPIC BOUND

In this section, we conduct numerical experiments to empirically analyze the EPIC bound. We plot the line charts of the EPIC bound $u_{\mathrm{EPIC}}(M; L)$ and the log-EPIC bound $\log(u_{\mathrm{EPIC}}(M; L))$ with respect to the number of student layers M on `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, as shown in Figure 1. We observe that $u_{\mathrm{EPIC}}(M; L)$ decreases rapidly when M is small, and then remains at a low level. Besides, we observe that $\log(u_{\mathrm{EPIC}}(M; L))$ decreases linearly, which implies that $u_{\mathrm{EPIC}}(M; L)$ converges exponentially to zero (note that $u_{\mathrm{EPIC}}(L, L) = 0$) as M increases. This empirically guarantees that we can distill deep GNNs into shallow GNNs.

6.4 ABLATION STUDIES

In this section, we conduct ablation studies to analyze the improvements of performance brought by $\mathcal{L}_{\mathrm{SL}}$ and $\mathcal{L}_{\mathrm{EP}}$. As shown in Table 3, both $\mathcal{L}_{\mathrm{SL}}$ and $\mathcal{L}_{\mathrm{EP}}$ bring improvements of performance to the EPIC students. Specifically, compared to vanilla EPIC students, $\mathcal{L}_{\mathrm{SL}}$ improves the performance by 0.03, 0.12, and 0.25 on `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, respectively. On this basis, $\mathcal{L}_{\mathrm{EP}}$ brings improvements of performance by 0.12, 0.75, and 0.60 on the three datasets, respectively. These results illustrate the effectiveness of the feature distillation loss and its potential in combination with other distillation loss terms.

7 CONCLUSION

In this paper, we propose the first GNN KD framework that quantitatively analyzes the teacher-student expressive power gap, namely EPIC. We show that the minimum error of approximating the teacher's embeddings with the student's embeddings has an upper bound, which decreases rapidly w.r.t. the number of student layers. We empirically demonstrate that the upper bound converges exponentially to zero as the number of student layers increases. Moreover, we propose to narrow the range of tuning the number of student layers based on the upper bound, and use an expressive power gap-related loss to further encourage the student to generate embeddings similar to those of the teacher. Experiments on large-scale benchmarks demonstrate that EPIC can effectively reduce the numbers of layers of deep GNNs, while achieving comparable or superior performance.

8 REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide our codes in supplementary materials and the details of selection of hyperparameters in Appendix A.6. For theoretical results, we provide detailed proof in Appendix B.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=i80OPhOCVH2`.

Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.*, 24:130–1, 2023.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/chen20v.html`.

Sikai Chen, Jiqian Dong, Paul Ha, Yujie Li, and Samuel Labi. Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering*, 36(7):838–857, 2021.

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2769–2781, 2022.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/71ee911dd06428a96c143a0b135041a4-Paper.pdf`.

Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. On provable benefits of depth in training graph convolutional networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9936–9949. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/524265e8b942930fbbe8a5d979d29205-Paper.pdf`.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Mengya Gao, Yujun Wang, and Liang Wan. Residual error based knowledge distillation. *Neurocomputing*, 433:154–161, 2021.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Amur Ghose, Vincent Zhang, Yingxue Zhang, Dong Li, Wulong Liu, and Mark Coates. Generalizable cross-graph embedding for gnn-based congestion prediction. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2021.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/gilmer17a.html`.

Peixian Gong, Chunyu Wang, and Lihua Zhang. Mmpoint-gnn: Graph neural network with dynamic edges for human activity recognition through a millimeter-wave radar. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2021.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

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*, 2023a. URL `https://openreview.net/forum?id=SM7XkJouWHm`.

Zhichun Guo, Chunhui Zhang, Yujie Fan, Yijun Tian, Chuxu Zhang, and Nitesh V Chawla. Boosting graph neural networks via adaptive knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7793–7801, 2023b.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1025–1035, 2017.

William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Huarui He, Jie Wang, Zhanqiu Zhang, and Feng Wu. Compressing deep graph neural networks via adversarial knowledge distillation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 534–544, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Tackling oversmoothing for general graph convolutional networks. *arXiv preprint arXiv:2008.09864*, 2020.

Cuiying Huo, Di Jin, Yawen Li, Dongxiao He, Yu-Bin Yang, and Lingfei Wu. T2-gnn: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4339–4346, 2023.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

AJAY KUMAR JAISWAL, Peihao Wang, Tianlong Chen, Justin F Rousseau, Ying Ding, and Zhangyang Wang. Old can be gold: Better gradient flow can make vanilla-GCNs great again. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=mhp4wLwiAI-`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

Donsuk Lee, Yiming Gu, Jerrick Hoang, and Micol Marchetti-Bowick. Joint interaction and trajectory prediction for autonomous driving using graph neural networks. *arXiv preprint arXiv:1912.07882*, 2019.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Pan Li and Jure Leskovec. The expressive power of graph neural networks. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao (eds.), *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 63–98. Springer Singapore, Singapore, 2022.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 338–348, 2020.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/bb04af0f7ecaee4aae62035497da1387-Paper.pdf`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5191–5198, 2020.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Yun Peng, Byron Choi, and Jianliang Xu. Graph learning for combinatorial optimization: a survey of state-of-the-art. *Data Science and Engineering*, 6(2):119–141, 2021.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh V Chawla. Nosmog: Learning noise-robust and structure-aware mlps on graphs. *arXiv preprint arXiv:2208.10010*, 2022.

Yijun Tian, Shichao Pei, Xiangliang Zhang, Chuxu Zhang, and Nitesh V Chawla. Knowledge distillation on graphs: A survey. *arXiv preprint arXiv:2302.00219*, 2023.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=7UmjRGzp-A`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International Conference on Machine Learning*, pp. 23341–23362. PMLR, 2022.

Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022a.

Ziyi Wang, Chen Bai, Zhuolun He, Guangliang Zhang, Qiang Xu, Tsung-Yi Ho, Bei Yu, and Yu Huang. Functionality matters in netlist representation learning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 61–66, 2022b.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z Li. Knowledge distillation improves graph structure augmentation for graph neural networks. *Advances in Neural Information Processing Systems*, 35:11815–11827, 2022a.

Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z Li. Quantifying the knowledge in gnns for reliable distillation into mlps. *arXiv preprint arXiv:2306.05628*, 2023.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=FQOC5u-1egI`.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022c.

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*, pp. 5453–5462. PMLR, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Keyulu Xu, Mozhi Zhang, Stefanie Jegelka, and Kenji Kawaguchi. Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. In *International Conference on Machine Learning*, pp. 11592–11602. PMLR, 2021.

Bencheng Yan, Chaokun Wang, Gaoyang Guo, and Yunkai Lou. Tinygnn: Learning efficient graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1848–1856, 2020.

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting over-smoothing in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020a.

Cheng Yang, Jiawei Liu, and Chuan Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the web conference 2021*, pp. 1227–1237, 2021.

Chenxiao Yang, Qitian Wu, and Junchi Yan. Geometric knowledge distillation: Topology compression for graph neural networks. *Advances in Neural Information Processing Systems*, 35: 29761–29775, 2022.

Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7074–7083, 2020b.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=_IY3_4psXuf`.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=r9hNv76KoT3`.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In *Neural Information Processing Systems*, 2020a. URL `https://api.semanticscholar.org/CorpusID:239998439`.

Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727*, 2021.

Wentao Zhang, Xupeng Miao, Yingxia Shao, Jiawei Jiang, Lei Chen, Olivier Ruas, and Bin Cui. Reliable data distillation on graph convolutional network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 1399–1414, 2020b.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *Advances in neural information processing systems*, 33:4917–4928, 2020.

Kuangqi Zhou, Yanfei Dong, Kaixin Wang, Wee Sun Lee, Bryan Hooi, Huan Xu, and Jiashi Feng. Understanding and resolving performance degradation in deep graph convolutional networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2728–2737, 2021.

Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, pp. 2848–2857, 2021.

A MORE DETAILS ABOUT EXPERIMENTS

In this section, we introduce more details about our experiments, including datasets, the inductive setting, teacher models, implementation details of baselines, and hyperparamters.

A.1 DATASETS

We evaluate our proposed EPIC on three OGB (Hu et al., 2020) large-scale datasets, i.e., `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`. Table 4 shows the statistics of the three datasets. Details about the datasets are as follows.

- `ogbn-arxiv` is a graph representing the citation network between all Computer Science arXiv papers indexed by MAG (Wang et al., 2020). Each node is an arXiv paper and each edge indicates that one paper cites another one. Each paper has a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. The embeddings of individual words are computed by running the skip-gram model (Mikolov et al., 2013) over the MAG corpus. The task is to predict the 40 subject areas of arXiv CS papers. The data split of `ogbn-arxiv` is realistic, which is based on the publication dates of the papers.

  In the stage of data preprocessing, we follow (Li et al., 2021) to convert the graph into an undirected graph and add self-loops to it.

- `ogbn-products` is a graph representing an Amazon product co-purchasing network. Nodes represent products sold in Amazon, and edges between two products indicate that the products are purchased together. Node features are generated by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100. The task is to predict the category of a product in a multi-class classification setup, where the 47 top-level categories are used for target labels. The dataset splitting of `ogbn-products` is challenging and realistic, as the split is based on the sales ranking (popularity). Specifically, the products are sorted according to their sales ranking. After that, the top 8%, next top 2%, and the rest are used for training, validation, and testing, respectively.

  In the stage of data preprocessing, we follow (Li et al., 2021) to add self-loops to the graph.

- `ogbn-proteins` is a graph representing a protein interaction network. Nodes represent proteins, and edges indicate different types of biologically meaningful associations between proteins. All edges com with 8-dimensional features, where each dimension represents the approximate confidence of a single association type and takes values between 0 and 1 (the larger the value is, the more confident we are about the association). The proteins come from 8 species. The task is to predict the presence of protein functions in a multi-label binary classification setup, where there are 112 kinds of labels to predict in total. The data split is based on the species, which the proteins come from.

  In the stage of data preprocessing, we follow (Li et al., 2021) to initialize node features through aggregating connected edge features by a sum aggregation.

Table 4: **Statistics of the three large-scale datasets.**

Datasets	**#Nodes**	**#Edges**	**Node Feat.**	**Edge Feat.**	**#Classes**	**Split Ratio**	**Metric**
`ogbn-arxiv`	169,343	1,166,243	✓	-	40	54/18/28	Accuracy
`ogbn-products`	2,449,029	61,859,140	✓	-	47	8/2/90	Accuracy
`ogbn-proteins`	132,534	39,561,252	-	✓	112	65/16/19	ROC-AUC

A.2 INDUCTIVE SETTING

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node features $\mathbf{X}$ and ground truth labels $\mathbf{y}$, where $\mathcal{V} = \mathcal{V}^{\mathrm{L}} \sqcup \mathcal{V}^{\mathrm{U}}$ is the disjoint union of the labeled node set $\mathcal{V}^{\mathrm{L}}$ and the unlabeled node set $\mathcal{V}^{\mathrm{U}}$, we remove all edges connected to nodes in $\mathcal{V}^{\mathrm{U}}$ from $\mathcal{E}$ to form $\mathcal{E}^{\mathrm{L}}$. Then, we construct a graph $\mathcal{G}^{\mathrm{L}} = (\mathcal{V}^{\mathrm{L}}, \mathcal{E}^{\mathrm{L}})$ with node features $\mathbf{X}^{\mathrm{L}}$ and groundtruth labels $\mathbf{y}^{\mathrm{L}}$. For clarity, we summarize the inductive setting as follows.

- Training the teacher: the input graph is $\mathcal{G}^{\mathrm{L}}$, the supervisory signal is $\mathbf{y}^{\mathrm{L}}$, and the outputs are the teacher's predictions on $\mathcal{V}^{\mathrm{L}}$. We save the teacher's soft labels $\mathbf{z}_{\mathcal{V}^{\mathrm{L}}}$ and embeddings $(\mathbf{H}_{\mathrm{T}}^{(L)})_{\mathcal{V}^{\mathrm{L}}}$ as the knowledge for training the student, where L is the number of teacher layers.
- Evaluating the teacher: the input graph is $\mathcal{G}$ and the outputs are the teacher's predictions on $\mathcal{V}$. We evaluate the test performance by comparing the teacher's predictions on $\mathcal{V}^{\mathrm{U}}$ with $\mathbf{y}^{\mathrm{U}}$.
- Training the student: the input graph is $\mathcal{G}^{\mathrm{L}}$, the supervisory signals are $\mathbf{y}^{L}$, $\mathbf{z}_{\mathcal{V}^{\mathrm{L}}}$, and $(\mathbf{H}_{\mathrm{T}}^{(L)})_{\mathcal{V}^{\mathrm{L}}}$, and the outputs are the student's predictions on $\mathcal{V}^{\mathrm{L}}$.
- Evaluating the student: the input graph is $\mathcal{G}$ and the outputs are the student's predictions on $\mathcal{V}$. We evaluate the test performance by comparing the student's predictions on $\mathcal{V}^{\mathrm{U}}$ with $\mathbf{y}^{\mathrm{U}}$.

A.3 TEACHER MODELS

For `ogbn-products` and `ogbn-proteins`, we select the one with the highest ranking on the public leaderboard among GNNs with over 100 layers. For the relatively smaller `ogbn-arxiv`, we select the one with the highest ranking on the public leaderboard among GNNs with over 10 layers. Based on these criteria, we select RevGCN-Deep (with 28 layers) (Li et al., 2021), RevGNN-112 (with 112 layers) (Li et al., 2021), and RevGNN-Deep (with 1,001 layers) (Li et al., 2021) as our teacher models on `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, respectively.

In the original paper (Li et al., 2021), the three models are trained and evaluated in the transductive setting. We retrain them in the inductive setting with the same hyperparameters used in the original paper. Because training the deep teachers takes a long time (e.g., it takes about 14 days to train RevGNN-Deep with 1,001 layers on `ogbn-proteins`), we train each teacher for only once. Specifically, on `ogbn-arxiv`, we run full-batch training and inference; on `ogbn-products`, we run mini-batch training with the graph partitioned into ten subgraphs and full-batch inference; on `ogbn-proteins`, we run mini-batch training and inference with the graph partitioned into ten and five subgraphs, respectively. Considering the randomness of the mini-batch inference, we perform 50 mini-batch inferences with different random seeds on `ogbn-proteins`, and report the mean and standard deviation of the results.

The three deep teacher models are based on the grouped reversible residual connections, which brings memory complexity that is independent of the depths of GNNs. Specifically, they uniformly partition the input node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ into N groups $\langle \mathbf{X}_1, \ldots, \mathbf{X}_N \rangle$ across the channel dimension, where $\mathbf{X} \in \mathbb{R}^{n \times \frac{d_x}{N}}$. The grouped reversible GNN block operates on a group of inputs and produces a group of inputs and generates a group of outputs: $\langle \mathbf{X}_1, \ldots, \mathbf{X}_N \rangle \mapsto \langle \mathbf{X}'_1, \ldots, \mathbf{X}'_N \rangle$. The forward pass is defined as

$$\mathbf{X}'_0 = \sum_{i=2}^{N} \mathbf{X}_i, \tag{12}$$

$$\mathbf{X}'_i = h_{\mathbf{w}_i}(\mathbf{X}'_{i-1}, \mathbf{A}, \mathbf{E}) + \mathbf{X}_i,\ i \in [N], \tag{13}$$

where $(h_{\mathbf{w}_i})_{i=1}^{N}$ are GNN layer functions (e.g., GCNConv (Kipf & Welling, 2017), GATConv (Veličković et al., 2018), and GENConv (Li et al., 2020)) with parameters $(\mathbf{w}_i)_{i=1}^{N}$. The inverse of the forward pass is

$$\mathbf{X}_i = \mathbf{X}'_i - h_{\mathbf{w}_i}(\mathbf{X}'_{i-1}, \mathbf{A}, \mathbf{E}),\ i \in \{2, \ldots, N\}, \tag{14}$$

$$\mathbf{X}'_0 = \sum_{i=2}^{N} \mathbf{X}_i, \tag{15}$$

$$\mathbf{X}_1 = \mathbf{X}'_1 - h_{\mathbf{w}_1}(\mathbf{X}'_0, \mathbf{A}, \mathbf{E}). \tag{16}$$

Therefore, the grouped reversible GNNs only need to save the output node embeddings of the last GNN block in GPU memory for back-propagation, and the memory complexity of activations is $\mathcal{O}(nd_h)$.

A.4 IMPLEMENTATION DETAILS AND HYPERPARAMETER SELECTION OF BASELINES

We compare EPIC with three representative KD4GNN frameworks, i.e., LSP (Yang et al., 2020b), GLNN (Zhang et al., 2021), and NOSMOG (Tian et al., 2022). The implementation details and hyperparameter selection of the baselines are as follows.

- LSP: we produce its PyG (Fey & Lenssen, 2019)-based implementation based on the DGL (Wang et al., 2019)-based source codes in its GitHub repository, and use it to train 4-layer GCNs on the three OGB large-scale datasets in the inductive setting. The range for our grid search for the hyperparameters are: hidden dimensions $h_d \in \{128, 256\}$, learning rate $\eta \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, dropout ratio $p \in \{0.1, 0.2, \ldots, 0.9\}$, the weight coefficients of the soft label loss $\mathcal{L}_{\text{SL}}$ and LSP loss $\mathcal{L}_{\text{LSP}}$, i.e., $\lambda_{\text{SL}}, \lambda_{\text{LSP}} \in \{1, 10^{-1}, 10^{-2}\}$.

- GLNN: we train MLPs with the source code in its GitHub repository on the three large-scale datasets in the inductive setting. The range for our grid search for the hyperparameters are: hidden dimensions $h_d \in \{512, 1024, 2048\}$, learning rate $\eta \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, dropout ratio $p \in \{0.1, 0.2, \ldots, 0.9\}$, and the weight coefficient of the soft label loss $\mathcal{L}_{\text{SL}} \in \{1, 10^{-1}, 10^{-2}\}$.

- NOSMOG: we train MLPs with the source code in its GitHub repository on the three large-scale datasets in the inductive setting. For the Deepwalk (Perozzi et al., 2014) algorithm used to generate positional encodings, the walking length is 50, the number of walking is 5, and the number of walking iteration is 1. The size of walking embedding is 128, 64, 128 for `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, respectively. It takes nearly 27 minites, 6 hours, and more than 50 hours to generate positional encodings for `ogbn-arxiv`, `ogbn-proteins`, and `ogbn-products`. Because the test nodes in `ogbn-products` account for 90%, the overall inference time (including the time for generating positional encoding) of the MLP trained with NOSMOG in the inductive setting will far exceed its GNN teacher. Therefore, we do not report the results of NOSMOG on `ogbn-products`. The range of our grid search for the hyperparameters are: hidden dimensions $h_d \in \{128, 256\}$, learning rate $\eta \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, dropout ratio $p \in \{0.1, 0.2, \ldots, 0.9\}$, the weight of the soft label loss $\mathcal{L}_{\text{SL}}$, i.e., $\lambda_{\text{SL}} \in \{1, 10^{-1}, 10^{-2}\}$, the weight of the representational similarity loss $\mathcal{L}_{\text{RSD}}$, i.e., $\lambda_{\text{RSD}} \in \{1, 10^{-1}, \ldots, 10^{-7}\}$, the weight of the adversarial learning loss $\mathcal{L}_{\text{ADV}}$, i.e., $\lambda_{\text{ADV}} \in \{5 \cdot 10^{-1}, 5 \cdot 10^{-2}, 5 \cdot 10^{-3}\}$.

A.5 FAIRNESS OF COMPARISONS WITH GNNS-TO-MLPS FRAMEWORKS

Some may be concerned about the fairness of our comparisons with existing GNNs-to-MLPs frameworks. Our explanation is as follows.

First, the selection of student models is a critical factor for knowledge distillation (Mirzadeh et al., 2020; Gao et al., 2021). Existing GNNs-to-MLPs works imply the view that ”MLPs are enough for the distillation of GNNs” and have triggered an upsurge in the GNNs-to-MLPs distillation. However, we claim that this view does not apply to the distillation for deep GNNs, and call for more attention to turn to the research on distilling deep GNNs using GNNs as students. Therefore, we select two GNNs-to-MLPs frameworks as baselines to support our claim.

Second, we fully understand that the reason why some may think the comparison is unfair, i.e., our student models are more expressive than MLPs, leading to the difference in the performance of distillation. However, this is exactly the claim we want to support through our experiments.

A.6 SELECTION OF ADDITIONAL HYPERPARAMETERS OF EPIC

Following the deep teachers (Li et al., 2021), our learning rate is $\eta = 10^{-3}$, dropout ratios are 0.5, 0.5, and 10^{-3} on `ogbn-arxiv`, `ogbn-products`, and `ogbn-proteins`, respectively. We set the hidden dimensions d_S of the EPIC students to 256, 320, and 160 on the three datasets respectively. The ratios γ for selecting the number of student layers we use are 0.75, 0.15, and 0.001 on the three datasets, respectively. The number K of concatenated embeddings are 3, 1, and 60 on the three datasets respectively, i.e., on `ogbn-arxiv` and `ogbn-proteins` we concatenate all hidden embeddings of the EPIC students, and on `ogbn-products` we use only the final embeddings of the EPIC student. The range of our grid search for the weight coefficients are $\lambda, \mu \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$.

B MORE ABOUT THEORETICAL ANALYSIS

B.1 RELATIONSHIP OF LINEAR-BASED THEORY TO NON-LINEAR GNNS

We analyze the expressive power of GNNs from a linear perspective because this is a standard practice (Wang & Zhang, 2022; Xu et al., 2021), and that existing works (Xu et al., 2021; Thekumparampil et al., 2018) have demonstrated that the theoretical analysis under the linear assumption is applicable to general non-linear GNNs.

B.2 PROOF OF THEOREM 1

In this subsection, we give the proof of Theorem 1.

Proof. Because $\widehat{\mathbf{L}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$, we have

$$\begin{aligned}
\mathbf{H}_T^{(L)} &= (a_0\mathbf{I} + \cdots + a_L\widehat{\mathbf{L}}^L)\phi_{\mathbf{w}_T}(\mathbf{X}) \\
&= \mathbf{U}(a_0\mathbf{I} + \cdots + a_L\boldsymbol{\Lambda}^L)\mathbf{U}^\top\phi_{\mathbf{w}_T}(\mathbf{X}) \\
&= \mathbf{U}(\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_T}(\mathbf{X})
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{H}_S^{(M)} &= (b_0\mathbf{I} + \cdots + b_M\widehat{\mathbf{L}}^M)\phi_{\mathbf{w}_S}(\mathbf{X}) \\
&= \mathbf{U}(b_0\mathbf{I} + \cdots + b_M\boldsymbol{\Lambda}^M)\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X}) \\
&= \mathbf{U}(\sum_{k=0}^{M} b_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X}).
\end{aligned}$$

Hence we have

$$\begin{aligned}
\|\mathbf{H}_S^{(M)} - \mathbf{H}_T^{(L)}\|_F &= \left\|\mathbf{U}(\sum_{k=0}^{M} b_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X}) - \mathbf{U}(\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_T}(\mathbf{X})\right\|_F \\
&\leq \left\|\mathbf{U}(\sum_{k=0}^{M} b_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X}) - \mathbf{U}(\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X})\right\|_F \\
&\quad + \left\|\mathbf{U}(\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X}) - \mathbf{U}(\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\phi_{\mathbf{w}_T}(\mathbf{X})\right\|_F \\
&\leq \|\mathbf{U}\|_F \left\|\sum_{k=0}^{M} b_k\boldsymbol{\Lambda}^k - \sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k\right\|_F \|\mathbf{U}^\top\phi_{\mathbf{w}_S}(\mathbf{X})\|_F \\
&\quad + \left\|\mathbf{U}(\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k)\mathbf{U}^\top\right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X}) - \phi_{\mathbf{w}_T}(\mathbf{X})\|_F \\
&\leq n^2\left\|\sum_{k=0}^{M} b_k\boldsymbol{\Lambda}^k - \sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k\right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X})\|_F \\
&\quad + n^2\left\|\sum_{k=0}^{L} a_k\boldsymbol{\Lambda}^k\right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X}) - \phi_{\mathbf{w}_T}(\mathbf{X})\|_F.
\end{aligned}$$

Taking the minimum value of the left and right sides of the above inequality, we have

$$\begin{aligned}
e(M;L) \leq n^2 \cdot & \min_{(b_k)_{k=0}^{M}, \mathbf{w}_S} \left\| \sum_{k=0}^{M} b_k \mathbf{\Lambda}^k - \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X})\|_F \\
& + n^2 \cdot \min_{(b_k)_{k=0}^{M}, \mathbf{w}_S} \left\| \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X}) - \phi_{\mathbf{w}_T}(\mathbf{X})\|_F \\
& \leq n^2 \cdot \min_{(b_k)_{k=0}^{M}} \left(\left\| \sum_{k=0}^{M} b_k \mathbf{\Lambda}^k - \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X})\|_F \right) \Big|_{\mathbf{w}_S = \mathbf{w}_T} \\
& + n^2 \cdot \min_{(b_k)_{k=0}^{M}} \left(\left\| \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F \|\phi_{\mathbf{w}_S}(\mathbf{X}) - \phi_{\mathbf{w}_T}(\mathbf{X})\|_F \right) \Big|_{\mathbf{w}_S = \mathbf{w}_T} \\
& = n^2 \cdot \min_{(b_k)_{k=0}^{M}} \left\| \sum_{k=0}^{M} b_k \mathbf{\Lambda}^k - \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F \|\phi_{\mathbf{w}_T}(\mathbf{X})\|_F + 0 \\
& \leq n^2 \cdot \frac{C}{n^2} \cdot \min_{(b_k)_{k=0}^{M}} \left\| \sum_{k=0}^{M} b_k \mathbf{\Lambda}^k - \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F \\
& \triangleq u(M;L) \qquad (17)
\end{aligned}$$

Define

$$h_1^{(M)}(b_0, \ldots, b_M) \triangleq \left\| \sum_{k=0}^{M} b_k \mathbf{\Lambda}^k - \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F^2 .$$

We can see that for any $(b_0, \ldots, b_M)$, there exist $(b'_0, \ldots, b'_M, b'_{M+1}) = (b_0, \ldots, b_M, 0)$ such that

$$h_1^{(M+1)}(b'_0, \ldots, b'_{M+1}) = h_1^{(M)}(b_0, \ldots, b_M),$$

hence

$$\min_{b_0, \ldots, b_{M+1}} h_1^{(M+1)}(b_0, \ldots, b_{M+1}) \leq \min_{b_0, \ldots, b_M} h_1^{(M)}(b_0, \ldots, b_M).$$

Then we have

$$u^L(M+1) \leq u^L(M), \qquad (18)$$

which means that $u^L(M)$ decreases monotonically with respect to M.

Our next step is to find $u(M;L)$. By letting $c_k = b_k - a_k$ for $k = 0, \ldots, M$, we have

$$\begin{aligned}
h_1^{(M)}(b_0, \ldots, b_M) &= \left\| \sum_{k=0}^{M} b_k \mathbf{\Lambda}^k - \sum_{k=0}^{L} a_k \mathbf{\Lambda}^k \right\|_F^2 \\
&= \sum_{i=1}^{n} \left(\sum_{k=0}^{M} (b_k - a_k) \lambda_i^k - \sum_{k=M+1}^{L} a_k \lambda_i^k \right)^2 \\
&= \sum_{i=1}^{n} \left(\sum_{k=0}^{M} c_k \lambda_i^k - \sum_{k=M+1}^{L} a_k \lambda_i^k \right)^2 \\
&\triangleq h_2^{(M)}(c_0, \ldots, c_M).
\end{aligned}$$

Then, the problem is transformed into solving

$$\min_{c_0, \ldots, c_M} h_2^{(M)}(c_0, \ldots, c_M),$$

which is equivalent to the least squares problem of

$$\min_{c_0, \ldots, c_M} \|\mathbf{V}^{(M)} \mathbf{c}^{(M)} - \mathbf{d}^{(M)}\|_F^2, \qquad (19)$$

where

$$\mathbf{V}^{(M)} = \begin{pmatrix} 1 & \lambda_1 & \cdots & \lambda_1^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_n & \cdots & \lambda_n^M \end{pmatrix}, \quad \mathbf{c}^{(M)} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_M \end{pmatrix}, \quad \mathbf{d}^{(M)} = \begin{pmatrix} \sum_{k=M+1}^{L} a_k \lambda_1^k \\ \vdots \\ \sum_{k=M+1}^{L} a_k \lambda_n^k \end{pmatrix}.$$

Let

$$\mathbf{V}^{(M)} = \begin{pmatrix} \mathbf{V}_{:,1}^{(M)} & \mathbf{V}_{:,2}^{(M)} & \cdots & \mathbf{V}_{:,M+1}^{(M)} \end{pmatrix}.$$

We first show that $\mathbf{V}^{(M)}$ has full column rank. Assume that there exist $(\alpha_k)_{k=1}^{M+1}$ that are not all zeros such that

$$\alpha_1 \mathbf{V}_{:,1}^{(M)} + \alpha_2 \mathbf{V}_{:,2}^{(M)} + \cdots + \alpha_{M+1} \mathbf{V}_{:,M+1}^{(M)} = 0,$$

which means that the M-degree polynomial with coefficients $(\alpha_k)_{k=1}^{M+1}$ has $n \gg M$ different roots. This leads to a contradiction, hence $\mathbf{V}^{(M)}$ has full column rank. And because $\operatorname{rank}(\mathbf{V}^{(M)}) \leq \min\{n, M+1\} = M+1$, we know that

$$\operatorname{rank}((\mathbf{V}^{(M)})^\top \mathbf{V}^{(M)}) = \operatorname{rank}(\mathbf{V}^{(M)}) = M+1,$$

which means that $(\mathbf{V}^{(M)})^\top \mathbf{V}^{(M)}$ is invertible. Therefore, the optimal solution to Problem (19) is

$$(\mathbf{c}^{(M)})^* = ((\mathbf{V}^{(M)})^\top \mathbf{V}^{(M)})^{-1} (\mathbf{V}^{(M)})^\top \mathbf{d}^{(M)} = (\mathbf{V}^{(M)})^\dagger \mathbf{d}^{(M)},$$

where $(\mathbf{V}^{(M)})^\dagger = ((\mathbf{V}^{(M)})^\top \mathbf{V}^{(M)})^{-1} (\mathbf{V}^{(M)})^\top$ is the Moore-Penrose inverse of $\mathbf{V}^{(M)}$. Thus, the minimum value is

$$\|\mathbf{V}^{(M)} (\mathbf{c}^{(M)})^* - \mathbf{d}^{(M)}\|_F^2 = \|\mathbf{V}^{(M)} (\mathbf{V}^{(M)})^\dagger \mathbf{d}^{(M)} - \mathbf{d}^{(M)}\|_F^2.$$

Suppose that the singular value decomposition (SVD) of $\mathbf{V}^{(M)}$ is $\mathbf{V}^{(M)} = \mathbf{P}^{(M)} \mathbf{\Sigma}^{(M)} (\mathbf{Q}^{(M)})^\top$, where $\mathbf{P}^{(M)} \in \mathbb{R}^{n \times (M+1)}$ and $\mathbf{Q}^{(M)} \in \mathbb{R}^{(M+1) \times n}$ satisfy $(\mathbf{P}^{(M)})^\top \mathbf{P}^{(M)} = (\mathbf{Q}^{(M)})^\top \mathbf{Q}^{(M)} = \mathbf{I}^{(M)}$, and $\mathbf{\Sigma}^{(M)} \in \mathbb{R}^{(M+1) \times (M+1)} = \operatorname{diag}(\sigma_0, \ldots, \sigma_M)$ is a diagonal matrix. Therefore, the SVD of $(\mathbf{V}^{(M)})^\dagger$ is $(\mathbf{V}^{(M)})^\dagger = \mathbf{Q}^{(M)} (\mathbf{\Sigma}^{(M)})^{-1} (\mathbf{P}^{(M)})^\top$, then we have $\mathbf{V}^{(M)} (\mathbf{V}^{(M)})^\dagger = \mathbf{P}^{(M)} (\mathbf{P}^{(M)})^\top$ and

$$\|\mathbf{V}^{(M)} (\mathbf{c}^{(M)})^* - \mathbf{d}^{(M)}\|_F^2 = \|\mathbf{P}^{(M)} (\mathbf{P}^{(M)})^\top \mathbf{d}^{(M)} - \mathbf{d}^{(M)}\|_F^2.$$

Hence, by Eq. (17), we know that

$$e(M; L) \leq C \|\mathbf{P}^{(M)} (\mathbf{P}^{(M)})^\top \mathbf{d}^{(M)} - \mathbf{d}^{(M)}\|_F.$$

□

C MORE EXPERIMENTS

C.1 TRAINING GNNS WITH FEWER LAYERS FROM SCRATCH

We conduct experiments of training student GNNs with fewer layers from scratch on the three OGB datasets, as shown in Table 5.

C.2 EXPERIMENTS IN THE TRANSDUCTIVE SETTING

We conduct experiments in the transductive setting, as shown in Table 6.

C.3 DISTILLATION FOR SHALLOW GNN TEACHERS

We conduct experiments of distilling shallow GNN teachers on `ogbn-arxiv` in the transductive and inductive settings, as shown in Table 7. Specifically, in the two settings, we train teacher models with 7 and 3 layers, and distill them to 3-layer and 1-layer students, and a 1-layer student, respectively.

Table 5: **Results of training student GNNs with fewer layers from scratch on the three OGB datasets.**

#Layers		2	3	4	5	6	7
`Arxiv`	w/ KD	72.57	73.06	73.18	73.24	73.35	73.24
	w/o KD	72.25	72.91	73.00	72.65	72.34	72.15
#Layers		2	3	4	5	6	7
`Products`	w/ KD	76.80	76.99	77.70	78.32	78.26	78.58
	w/o KD	76.67	76.86	76.85	77.29	77.56	77.71
#Layers		10	20	30	40	50	60
`Proteins`	w/ KD	85.64	85.68	85.33	85.81	85.72	85.85
	w/o KD	84.13	84.23	84.58	84.75	84.89	85.00

Table 6: **Results in the transductive setting.**

Datasets	**Teacher**	**LSP**	**GLNN**	**NOSMOG**	**EPIC**
`Arxiv`	72.96	69.77	57.41	65.79	73.33
`Products`	82.11	78.31	61.45	64.62	83.73
`Proteins`	87.42	-	75.19	73.84	87.13

Table 7: **Results of distilling shallow GNN teachers on `ogbn-arxiv`.**

Setting	**Teacher-7layers**	**EPIC-3layers**	**EPIC-1layer**	**Teacher-3layers**	**EPIC-1layer**
Transductive	70.93	72.31	70.59	70.35	70.57
Inductive	70.69	71.91	70.42	69.97	70.13

Table 8: **Comparisons with BGNN in the transductive and inductive settings.**

Datasets	*Transductive*			*Inductive*		
	Teacher	**BGNN**	**EPIC**	**Teacher**	**BGNN**	**EPIC**
`Arxiv`	72.91	68.87	73.06	72.96	69.94	73.33
`Products`	77.86	74.01	78.58	82.11	79.45	83.73
`Proteins`	85.91	-	85.85	87.42	-	87.13

Table 9: **Comparisons with BGNN in the inductive setting.**

Setting	**Teacher-7layers**	**EPIC-3layers**	**EPIC-1layer**	**Teacher-3layers**	**EPIC-1layer**
Transductive	70.93	72.31	70.59	70.35	70.57
Inductive	70.69	71.91	70.42	69.97	70.13

C.4 MORE BASELINES

We compare EPIC with BGNN (Guo et al., 2023b), which uses 2-layer GCNs as students. We implement BGNN ourselves, as BGNN is not open source. The results are shown in Tables 8 and 9

D MOTIVATION FOR STUDYING THE DISTILLATION FOR DEEP GNNS

We summarize the motivation for studying the distillation for deep GNNs as follows.

First, on large-scale datasets (e.g., ogbn-arxiv, ogbn-products, and ogbn-proteins), the deep teacher models that we distill in our paper (i.e., RevGCN-Deep, RevGNN-112, and RevGNN-Deep) outperform what existing KD4GNN works aim to distill, such as GCN, GraphSAGE, GAT, and APPNP. Compared with shallow GNNs, deep GNNs with more message passing layers are better at capturing long-range patterns (Chen et al., 2022; Cong et al., 2021).

Second, although some deep GNNs still face challenges such as over-fitting and over-squashing, there have been many efforts to solve these challenges, such as (Li et al., 2021; Chen et al., 2020; Zhao & Akoglu, 2019; Guo et al., 2023a; Zhou et al., 2020; 2021; JAISWAL et al., 2022). We believe that the challenges are temporary obstacles to the development of deep GNNs, and we believe that deep GNNs will receive widespread attention in the future.

Last but not least, the slow inference speed is also one of the important factors limiting the real applications of deep GNNs. Despite the great success in compressing and accelerating moderate-sized or shallow GNNs of existing KD techniques, it still remains a tough challenge to distill deep GNNs due to the huge expressive power gap between the teachers and students. Therefore, this paper aims to address the challenge of distilling deep GNNs that excel on large-scale graphs and possess numerous layers.