On Self-Distilling Graph Neural Network

Yuzhao Chen[1,2,*], **Yatao Bian**[2,†], **Xi Xiao**[1,3,†], **Yu Rong**[2], **Tingyang Xu**[2], **Junzhou Huang**[2,4]
[1]Tsinghua University
[2]Tencent AI Lab
[3]Peng Cheng Laboratory
[4]University of Texas at Arlington

Abstract

Recently, the teacher-student knowledge distillation framework has demonstrated its potential in training Graph Neural Networks (GNNs). However, due to the difficulty of training over-parameterized GNN models, one may not easily obtain a satisfactory teacher model for distillation. Furthermore, the inefficient training process of teacher-student knowledge distillation also impedes its applications in GNN models. In this paper, we propose the *first* teacher-free knowledge distillation method for GNNs, termed GNN Self-Distillation (GNN-SD), that serves as a drop-in replacement of the standard training process. The method is built upon the proposed neighborhood discrepancy rate (NDR), which quantifies the non-smoothness of the embedded graph in an efficient way. Based on this metric, we propose the adaptive discrepancy retaining (ADR) regularizer to empower the transferability of knowledge that maintains high neighborhood discrepancy across GNN layers. We also summarize a generic GNN-SD framework that could be exploited to induce other distillation strategies. Experiments further prove the effectiveness and generalization of our approach, as it brings: 1) state-of-the-art GNN distillation performance with less training cost, 2) consistent and considerable performance enhancement for various popular backbones.

1 Introduction

Knowledge Distillation (KD) has demonstrated its effectiveness in boosting compact neural networks. Yet, most of the KD researches focus on Convolutional Neural Networks (CNNs) with regular data as input instances, while little attention has been devoted to Graph Neural Networks (GNNs) that are capable of processing irregular data. A significant discrepancy is that GNNs involve the topological information into the updating of feature embeddings across network layers, which is not taken into account in the existing KD schemes, restricting their potential extensions to GNNs.

A recent work, termed LSP [2020], proposed to combine KD with GNNs by transferring the local structure, which is modeled as the distribution of the similarity of connected node pairs, from a pre-trained teacher GNN to a light-weight student GNN. However, there exists a major concern on the selection of qualified teacher GNN models. On the one hand, it's likely to cause performance degradation once improper teacher networks are selected [2019]. On the another hand, the performance of GNNs is not always indicated by their model capacity due to the issues of over-smoothing [2018] and over-fitting [2019], which have caused obstacles to train over-parameterized and powerful GNNs. As a result, every time encountering a new learning task, one may spend substantial efforts in searching for a qualified GNN architecture to work as a teacher model, and thus the generalization ability of this method remains a challenge. Another barrier of the adopted conventional teacher-student framework is the inefficiency in the training process. Such distillation pipeline usually requires two steps: first, pre-training a relatively heavy model, and second, transferring the forward predictions (or transformed features) of the teacher model to the student model. With the assistance of the teacher model, the training cost would tremendously increase more than twice than an ordinary training procedure.

In this work, we resort to cope with these issues via the self-distillation techniques (or termed teacher-free distillation), which perform knowledge extraction and transfer between layers of a single network without the assistance from auxiliary models [2019; 2019]. Our work provides the first dedicated self-distillation approach designed for generic GNNs, named GNN-SD. The core ingredient of GNN-SD is motivated by the mentioned challenge of over-smoothing, which occurs when GNNs go deeper and lead the node features to lose their discriminative power. Intuitively, one may avoid such dissatisfied cases by pushing node embeddings in deep layers to be distinguishable from their neighbors, which is exactly the property possessed by shallow GNN layers.

To this end, we first present the *Neighborhood Discrepancy Rate* (NDR) to serve as an approximate metric in quantifying the non-smoothness of the embedded graph at each GNN layer. Under such knowledge refined by NDR, we propose to perform knowledge self-distillation by an adaptive dis-

[*]This work is done when Yuzhao Chen works as an intern in Tencent AI Lab

[†]Corresponding authors: Yatao Bian and Xi Xiao

crepancy retaining (ADR) regularizer. The ADR regularizer adaptively selects the target knowledge contained in shallow layers as the supervision signal and retains it to deeper layers. Furthermore, we summarize a generic GNN-SD framework that could be exploited to derive other distillation strategies. As an instance, we extend GNN-SD to involve another knowledge source of *compact graph embedding* for better matching the requirements of graph classification tasks. In a nutshell, our main contributions are:

- We present GNN-SD, to our knowledge, the first generic framework designed for distilling the graph neural networks with no assistance from extra teacher models. It serves as a drop-in replacement of the standard training process to improve the training dynamics.
- We introduce a simple yet efficient metric of NDR to refine the knowledge from each GNN layer. Based on it, the ADR regularizer is proposed to empower the adaptive knowledge transfer inside a single GNN model.
- We validate the effectiveness and generalization ability of our GNN-SD by conducting experiments on multiple popular GNN models, yielding the state-of-the-art distillation result and consistent performance improvement against baselines.

2 Related Work

Graph Neural Network Recently, Graph Neural Networks (GNNs), which propose to perform message passing across nodes in the graph and updating their representation, has achieved great success on various tasks with irregular data, such as node classification, protein property prediction to name a few. Working as a crucial tool for graph representation learning, however, these models encounter the challenge of over-smoothing. It says that the representations of the nodes in GNNs would converge to a stationary point and become indistinguishable from each other when the number of layers in GNNs increases. This phenomenon limits the depth of the GNNs and thus hinders their representation power.

One solution to alleviate this problem is to design network architectures that can better memorize and utilize the initial node features. Representative papers includes GCN [2016], JKNet [2018b] DeeperGCN [2020], GCNII [2020b], etc. On the other hand, methods like DropEdge [2019] and AdaEdge [2020a] have proposed solutions from the view of conducting data augmentation.

In this paper, we design a distillation approach tailored for GNNs, which also provides a feasible solution to this problem. Somewhat related, Chen *et al.* [2020a] proposed a regularizer to the training loss, which simply forces the nodes in the last GNN layer to obtain a large distance between remote nodes and their neighboring nodes. However, it can only obtain a slight performance improvement.

Teacher-Student Knowledge Distillation Knowledge distillation [2015], aims at transferring the knowledge hidden in the target network (i.e. teacher model) into the online network (i.e. student model) that is typically light-weight, so that the student achieves better performance compared with the one trained in an ordinary way. Generally, there exist two technical routes for KD. The first one is closely related to label smoothing [2020], which utilizes the output distribution of the teacher model to serve as a smooth label for training the student. Another line of research is termed as feature distillation [2014; 2016; 2018], which exploits the semantic information contained in the intermediate representations. As summarized in [2019], with different concerning knowledge to distill, these methods can be distinguished by the formulation of feature transformation and knowledge matching loss function.

Recently, Yang *et al.* [2020] studied the teacher-student distillation methods in training GNNs. They extract the knowledge of local graph structure based on the similarity of connected node pairs from the teacher model and student model, then perform distillation by forcing the student model to match such knowledge. However, the performance improvement resulted from such conventional distillation framework does not come with a free price, as discussed in Section 1.

Self-Distillation For addressing the issues of the teacher-student framework, a new research area termed teacher-free distillation, or self-distillation, attracts a surge of attention recently. Throughout this work, we refer this notion to the KD techniques that perform knowledge refining and transfer between network layers inside a single model. In this way, the distillation learning could be conducted with a single forward propagation in each training iteration. BYOT [2019] proposed the first self-distillation method. They consider that the teacher and student are composed in the same networks since the deeper part of the networks can extract more semantic information than the shallow one. Naturally, they manage to distill feature representations as well as the smooth label from deeper layers into the shallow layers. Similarly, Hou *et al.* [2019] proposed to distill the attention feature maps from the deep layers to shallow ones for lane detection. However, these methods focus on the application of CNNs, neglecting the usage of graph topology information, and thus restricting their potential extension to GNNs.

3 GNN Self-Distillation

A straightforward solution to perform self-distillation for training GNNs is to supervise the hidden states of shallow layers by the ones of deep layers as the target, as the scheme proposed in BYOT [2019]. However, we empirically find that such a strategy leads to performance degradation. It's on the one hand attributed to the neglection of the graph topological information. On the other hand, it's too burdensome for shallow GNN layers to match their outputs to such unrefined knowledge and eventually leads to over-regularizing. Furthermore, such a simple solution requires fixed representation dimensionalities across network layers, which limits its applications to generic GNN models. In the following sections, we first introduce the key ingredient of our GNN Self-Distillation (GNN-SD). Then we summarize a unified GNN-SD framework that is well extendable to other distillation variants.

Notation Throughout this work, a graph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, where $\mathcal{V}$ is vertex set that has N nodes with d-dimension features of $\mathbf{X} \in \mathbb{R}^{N \times d}$, edge set $\mathcal{E}$ of size M

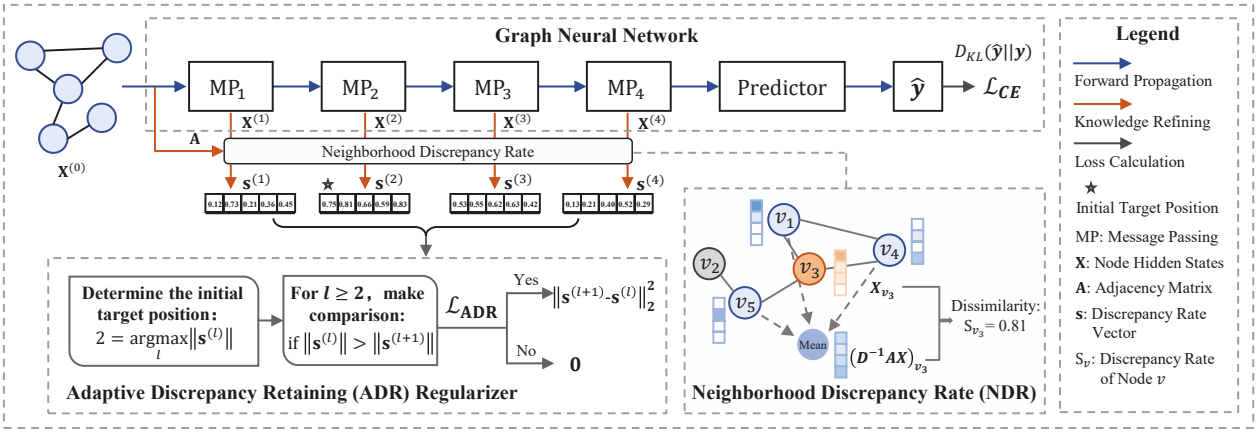

Figure 1: The schemata of the distillation strategy of adaptive discrepancy retaining. A four-layer GNN is adopted for the illustration.

is encoded with edge features of $\mathbf{E}$, and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix. The node degree matrix is given by $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_N)$. Node hidden states of l-th GNN layer is denoted as $\mathbf{X}^{(l)}$, and the initial hidden states $\mathbf{X}^{(0)}$ is usually set as the node intrinsic features $\mathbf{X}$. Given a node v, its connected-neighbors are denoted as $\mathcal{N}_v$. For a matrix $\mathbf{X}$, $\mathbf{X}_{i\cdot}$ denotes its i-th row and $\mathbf{X}_{\cdot j}$ denotes its j-th column.

3.1 Adaptive Discrepancy Retaining

Since the well-known over-smoothing issue of GNNs occurs when the input graph data flows into deep layers, an inspired insight is that we can utilize the property of non-smoothness in shallow GNN layers, and distill such knowledge into deep ones. In this way, the model is self-guided to retain non-smoothness from the initial embedded graph to the final embedded output. The remained questions are: how to refine the desired knowledge from the fine-grained node embeddings? and what is a proper knowledge transfer strategy?

Neighborhood Discrepancy Rate To answer the first question, we introduce the module of Neighborhood Discrepancy Rate (NDR), which is used to quantify the non-smoothness of each GNN layer.

It's proved that nodes in a graph component converge to the same stationary point while iteratively updating the message passing process of GNNs, and hence the hidden states of connected-nodes become indistinguishable [2018]. It implies that, given a node v in the graph,

$$\sum\nolimits_{c\in\mathcal{N}_v} \|\mathbf{X}^{(l)}_{v\cdot} - \mathbf{X}^{(l)}_{c\cdot}\|_p < \epsilon, \text{if } l \to \infty, \forall \epsilon > 0. \quad (1)$$

As a result, it leads to the issue of over-smoothing and further hurts the accuracy of node classification. Centering around this conclusion, it is a natural choice to use the pair-wise metric (and the resulting distance matrix) to define the local non-smoothness of the embedded graph. However, such fine-grained knowledge might still cause over-regularization for GNNs trained under teacher-free distillation. By introducing the following proposition (details deferred to Appendix), one can easily derive from formula (1) that, $\|\mathbf{X}^{(l)}_{v\cdot} - (\mathbf{D}^{-1}\mathbf{A}\mathbf{X})^{(l)}_{v\cdot}\|_p < \epsilon$ as layer l goes to infinity.

Proposition 1 *Suppose $d_1(\mathbf{X}, \mathcal{G})$ calculates the L_p-norm of the difference of each central node and their aggregated neighbors in the graph $\mathcal{G}$, and the pair-wise distance metric $d_2(\mathbf{X}, \mathcal{G})$, on the other hand, computes the difference at the level of node pairs. Then, it holds: $d_2(\mathbf{X}, \mathcal{G}) \geq d_1(\mathbf{X}, \mathcal{G})$.*

We leverage this property to refine the knowledge from a higher level. Specifically, given a central node v in layer l, we first obtain the aggregation of its adjacent nodes to work as the virtual node that indicates its overall neighborhood, $\mathbf{N}^{(l)}_v = (\mathbf{D}^{-1}\mathbf{A}\mathbf{X}^{(l)})_{v\cdot}$. For excluding the effect of embeddings' magnitude, we use the cosine similarity between the embeddings of central node $\mathbf{X}^{(l)}_{v\cdot}$ and virtual node $\mathbf{N}^{(l)}_v$ to calculate their affinity, and transform it into a distance metric,

$$S^{(l)}_v = 1 - \frac{\mathbf{X}^{(l)}_{v\cdot}(\mathbf{A}\mathbf{X}^{(l)})^{\mathrm{T}}_{v\cdot}}{||\mathbf{X}^{(l)}_{v\cdot}||_2 \cdot ||(\mathbf{A}\mathbf{X}^{(l)})_{v\cdot}||_2}, v = 1, ..., N, \quad (2)$$

Note that it's not needed to perform the inverse matrix multiplication of node degrees due to the normalization conducted by cosine similarity metric. The defined $S^{(l)}_v$ of all nodes compose the neighborhood discrepancy rate of layer l:

$$\mathbf{s}^{(l)} = (S^{(l)}_1, ..., S^{(l)}_N). \quad (3)$$

Compared with the pair-wise metric, the NDR extracts neighbor-wise non-smoothness, which is easier to transfer and prevents over-regularizing by self-distillation. Moreover, it can be easily implemented with two consecutive matrix multiplication operations, enjoying a significant computational advantage. The NDR also possesses better flexibility to model local non-smoothness of the graph, since pair-wise metrics can not be naturally applied together with layer-wise sampling techniques [2018; 2018].

Specially, for the task of node classification, there is another reasonable formulation of the virtual neighboring node. That is, taking node labels into account, $\mathbf{N}^{(l)}_v = (\mathbf{D}^{-1}\mathbf{A}'\mathbf{X}^{(l)})_{v\cdot}$, where $\mathbf{A}' = \mathbf{A} \odot \mathbf{Y}$ denotes the masked adjacency, $\mathbf{Y} \in \mathbb{R}^{N\times N}$ the binary matrix with entries $\mathbf{Y}_{i,j}$ equal to 1 if node i and j are adjacent and belong to different categories, and $\odot$ the element-wise multiplication operator.

Then, the NDR would not count the discrepancy of nodes that are supposed to share high similarity. For unity and simplicity, we still use the former definition throughout this work.

Strategy for Retaining Neighborhood Discrepancy Previous self-distillation methods usually treat the deep representations as the target supervision signals, since they are considered to contain more semantic information. However, we found that such a strategy is not optimal, sometimes even detrimental for training GNNs (refer to Appendix for details). The rationale behind our design of distilling neighborhood discrepancy is to retain the non-smoothness, which is extracted as the knowledge by NDR, from shallow GNN layers to the deep ones. In details, we design the following guidelines (refer to Appendix for more analysis) for the self-distillation learning, which formulate our adaptive discrepancy retaining (ADR) regularizer:

- The noise in shallow network layers might cause the calculated NDR of the first few embedded graphs to be inaccurate, thus the initial supervision target is adaptively determined by the magnitude of the calculated NDR.
- For facilitating the distillation learning, the knowledge transfer should be progressive. Hence, the ADR loss is computed by matching the NDR of deep layer (online layer) to the target one of its previous layer (target layer).
- Explicit and adaptive teacher selection is performed, i.e the ADR regularizes the GNN only when the magnitude of NDR of the target layer is larger than the online layer.
- Considering that the nodes in regions of different connected densities have different rates of becoming over-smoothing [2018], the matching loss can be weighted by the normalized node degrees (denoted as $\mathbf{D}$ for conciseness) to emphasize such a difference.

As a result, the final ADR regularizer is defined as:

$$\mathcal{L}_{\mathrm{N}} = \sum_{l=l^{\star},...,L-1} \mathbb{1}(\|\mathbf{s}^{(l)}\| > \|\mathbf{s}^{(l+1)}\|) d^2(\mathbf{s}^{(l+1)}, \mathbf{s}^{(l)}), \quad (4)$$

where the indicator function $\mathbb{1}(\cdot)$ performs the teacher selection, $l^{\star} = \mathrm{argmax}_k\{\|\mathbf{s}^{(k)}\| | k \in \{1, ..., L-1\}\}$ determine the position of initial supervision target, and $d^2(\mathbf{s}^{(l+1)}, \mathbf{s}^{(l)}) = \|\mathbf{D}(\mathbf{s}^{(l+1)} - \mathrm{SG}(\mathbf{s}^{(l)}))^{\mathrm{T}}\|_2^2$ is the degree-weighted mean squared error function that calculates the knowledge matching loss. Here $\mathrm{SG}(\cdot)$ denotes the Stop Gradient operation, meaning that the gradient of the target NDR tensor is detached in the implementation, for serving as a supervision signal. The approach is depicted in Figure 1.

We also analytically demonstrate that the proposed discrepancy retaining can be comprehended from the perspective of information theory. This is analogous to the concept in [2019]. Specifically, the retaining of neighborhood discrepancy rate encourages the online layer to share high mutual information with the target layer, as illustrated in the following proposition (I stands for mutual information, H denotes the entropy, details are deferred to Appendix).

Proposition 2 *The optimization of the ADR loss increase the lower bound of the mutual information between the target NDR and the online one. That is, the inequality holds:* $I(\mathbf{s}^{(l)}, \mathbf{s}^{(l+1)}) \geq H(\mathbf{s}^{(l)}) - \mathbb{E}_{\mathbf{s}^{(l)}, \mathbf{s}^{(l+1)}}[\|\mathbf{D}(\mathbf{s}^{(l+1)} - \mathbf{s}^{(l)})^T\|_2^2]$.

3.2 Generic GNN-SD Framework

Generally, by refining and transferring compact and informative knowledge between layers, self-distillation on GNNs can be summarized as the learning of the additional mapping,

$$\mathcal{M}_{\mathrm{SD}}^{g,L} : \mathcal{T}_s(\mathcal{C}_s(\mathcal{G}, P_s)) \rightarrow \mathcal{T}_t(\mathcal{C}_t(\mathcal{G}, P_t)), \quad (5)$$

where $P \in \{1, ..., L\}$ is the layer position to extract knowledge from the network, $\mathcal{C}$ denotes the granularity (or coarseness) of the embedded graph, $\mathcal{T}$ represents the specific transformation applied to the chosen embeddings, and the subscripts of s and t denote the identity of student (to simulate) and teacher (to transfer), respectively.

Naturally, the combinations of different granularities and transformation functions lead to various distilled knowledge. As an instance, we show here to involve another knowledge source of the full-graph embedding, and provide further discussions in Appendix for completeness.

Considering the scenario of graph classification, where GNNs might focus more on obtaining meaningful embedding of the entire graph than individual nodes, the full-graph embedding could be the well-suited knowledge, since it provides a global view of the embedded graph (while the NDR captures the local property),

$$\mathcal{C}(\mathcal{G}, \mathcal{P}) := \mathbf{G}^{(P)} = \mathrm{Readout}_{v \in \mathcal{G}}(\mathbf{X}_{v\cdot}^{(P)}), \quad (6)$$

where $\mathrm{Readout}$ is a permutation invariant operator that aggregates embedded nodes to a single embedding vector. In contrast to the fine-grained node features, the coarse-grained graph embedding is sufficiently compact so as we can simply use the identity function to preserve the transferred knowledge. Hence the target mapping is:

$$\mathcal{M}_{\mathrm{graph}}^{g,L} : \mathbb{1}(\mathbf{G}^{(l+1)}) \rightarrow \mathbb{1}(\mathbf{G}^{(l)}). \quad (7)$$

It can be learned by optimizing the graph-level distilling loss:

$$L_{\mathrm{G}} = \sum\nolimits_{l=1,...,L-1} ||\mathbf{G}^{(l+1)} - \mathrm{SG}(\mathbf{G}^{(l)})||_2^2. \quad (8)$$

In this way, GNN-SD extends the utilization of mixed knowledge sources over different granularities of the embedded graph.

Overall Loss The total loss function is formulated as:

$$\mathcal{L}_{\mathrm{T}} = \mathrm{CE}(\hat{y}, y) + \alpha \mathcal{L}_{\mathrm{L}} + \beta \mathcal{L}_{\mathrm{N}} + \gamma \mathcal{L}_{\mathrm{G}}. \quad (9)$$

The first term calculates the basic cross entropy loss between the final predicted distribution $\hat{y}$ and the ground-truth label y. The second term, borrowing from [2019], regularizes the intermediate logits generated by intermediate layers to mimic the final predicted distribution for accelerating the training and improving the capacity of shallow layers. We provide its formulation in Appendix to make it self-contained. The remaining terms are defined in Eq.(4) and Eq.(8). α, β, and γ are the hyper-parameters that balance the supervision of the distilling objectives and target label.

4 Experiments

4.1 Exploring Analysis of Discrepancy Retaining

We first conduct exploring analysis to investigate how the ADR regularizer helps improve the training dynamics of GNNs. Hyper-parameters of α and γ are fixed to 0.

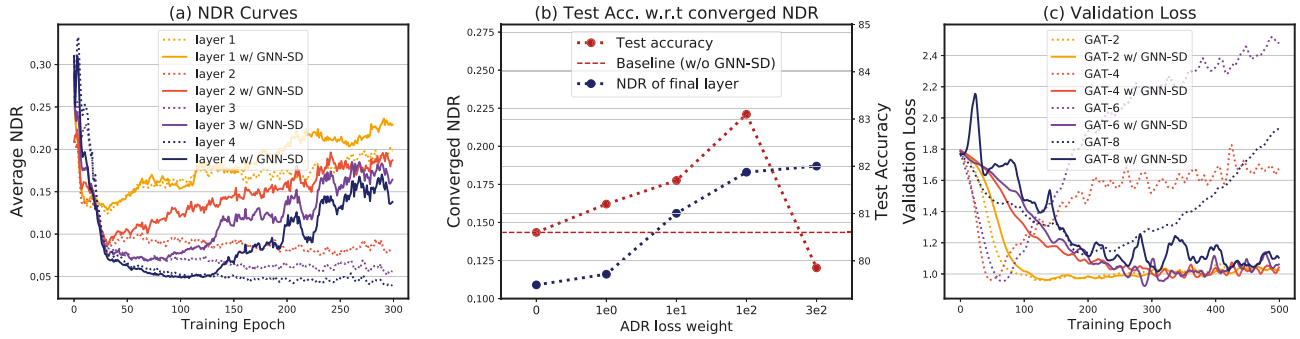

Figure 2: (a) Comparison of NDR between training w/ (solid line) and w/o (dotted line) imposing ADR regularizer. Smaller value means suffering from more over-smoothing. (b) The correlation between test accuracy and converged NDR of the final layer, controlled by the loss weights of ADR regularizer. (c) Comparison of validation loss curves. GAT-n denotes GAT model with n hidden layers.

In Figure 2(a), we show the comparison of NDR curves of each layer in a 4-layer GraphSage [2017]. In line with the expectations, the neighborhood discrepancy drops when the layer grows, and the final layer (in dark blue, dotted line) approaches a close-to-zero value of NDR, which indicates that the output node embeddings become indistinguishable with their neighbors. Conversely, the GNN trained with GNN-SD preserves higher discrepancy over shallow to deep layers, and even reveals an increasing tendency as the training progresses. It might imply that the GNN gradually learns to pull the connected-node embeddings away if they're not of the same class, for obtaining more generalized representations. This observation indicates ADR's effect on *alleviating the over-smoothing issue*. We provide related examples in Appendix that motivate the development of ADR regularizer.

In Figure 2(b), we study the correlation of model performance and the converged NDR: as the ADR loss weight increases in a reasonable range (e.g. from $1e0$ to $1e2$), the NDR increases and the performance gain improves (from 0.6% to 2.5%). The shown *positive correlation between test accuracy and NDR* verifies the rationality of the NDR metric and the distillation strategy as well. Notably, there also exists a trade-off in determining the optimal loss weight. If an overestimated weight ($1e3$) was assigned, the knowledge transfer task would become too burdensome for the GNN model to learn the main classification task and hurt the performance.

Figure 2(c) depicts the validation loss on GAT [2018] backbones with different depths. The validation loss curves are dramatically pulled down after applying GNN-SD. It explains that ADR regularizer also helps GNNs *relieve the issue of over-fitting*, which is known as another tough obstacle in training deep GNNs.

4.2 Comparison with KD Methods

We compare our method with other distillation methods, including AT [2016], FitNet [2014], BYOT and LSP. We follow [2020] to perform the comparisons on the baseline of a 5-layer GAT on the PPI dataset [2017]. For evaluating AT and FitNet in a teacher-free KD scheme, we follow their papers to perform transformations on node embeddings to get the attention maps and intermediate features, respectively. We also evaluate BYOT to see the effect of intermediate logits.

Method	Knowledge Source	F1 Score	Time
Teacher (T)	/	97.6	0.85s
Baseline	/	95.7	**0.62s**
AT	$T(\mathbf{X})$	95.4	1.75s
FitNet	$T(\mathbf{X})$	95.6	1.99s
LSP	$T(\mathbf{X})$,$T(\mathbf{A})$	96.1	1.90s
Baseline	/	$95.61_{\pm 0.20}$	**0.62s**
AT	$\mathbf{X}$	$95.88_{\pm 0.25}$	0.73s
FitNet	$\mathbf{X}$	$95.60_{\pm 0.17}$	0.95s
BYOT	$\mathbf{X}$	$95.81_{\pm 0.56}$	0.80s
GNN-SD	$\mathbf{X}$,$\mathbf{A}$	$\mathbf{96.20}_{\pm 0.03}$	0.87s

Table 1: Performance comparison with other distillation methods. The second column indicates the knowledge source, $\mathbf{X}$ is node features, $\mathbf{A}$ is the adjacency matrix, $T(\cdot)$ denotes the teacher model.

The experiments are conducted for 4 runs, and those under teacher-student framework are cited from [2020]. For our method, the hyper-parameters of α and β are both set to 0.01 and γ is 0. Results are summarized in Table 1. Clearly, GNN-SD obtains significant performance promotion against other self-distillation methods by involving the topological information into the refined knowledge. GNN-SD also achieves a better performance gain (0.59) on the baseline compared with LSP (0.4), even when our method does not require the assistance from an extra teacher model. In this way, the training cost is greatly saved (40% training time per epoch increase against baseline for GNN-SD v.s. 190% increase for LSP). Besides, it avoids the process of selecting qualified teachers, bringing much better usability.

4.3 Overall Comparison Results

Node Classification Table 2 summarizes the results of GNNs with various depths on Cora, Citeseer and PubMed [2008]. We follow the setting of semi-supervised learning, using the fixed 20 nodes per class for training. The hyper-parameters of γ is fixed to 0 for node classification, and we determine α and β via a simple grid search. Details are provided in Appendix. It's observed that GNN-SD consistently improves the test accuracy for all cases. Generally, GNN-SD yields larger improvement for deeper architecture, as it gains 0.5% average improvement for a two-layer GAT on Cora while achieving 3.2% increase for the 8-layer one.

Dataset	Model	Layers			
		2	4	8	16
Cora	GAT	83.2	80.1	76.9	74.8
	GAT w/ GNN-SD	**83.7**	81.2	**80.1**	77.6
	GraphSage	81.3	80.3	78.8	77.2
	GraphSage w/ GNN-SD	81.7	**81.5**	79.8	**78.2**
Citeseer	GAT	72.5	70.5	65.1	64.5
	GAT w/ GNN-SD	72.6	**71.5**	**68.3**	**66.2**
	GraphSage	72.3	70.7	61.7	59.2
	GraphSage w/ GNN-SD	**72.7**	71.0	64.5	61.8
Pubmed	GAT	79.2	78.5	76.6	75.6
	GAT w/ GNN-SD	**79.5**	**79.4**	**78.5**	76.6
	GraphSage	78.8	77.9	73.8	77.2
	GraphSage w/ GNN-SD	79.2	79.4	77.6	**78.2**

Table 2: Node classification on varying-depths models.

Graph Classification Table 3 summarizes the results of various popular GNNs on the graph kernel classification datasets, including ENZYMES, DD, and PROTEINS in TU dataset [2016]. Since there exist no default splittings, each experiment is conducted by 10-fold cross validation with the splits ratio at 8:1:1 for training, validating and testing. We choose five widely-used GNN models, including GCN, GAT, GraphSage, GIN [2018a] and GatedGCN [2017], to work as the evaluation baselines. Hyper-parameter settings are deferred to Appendix. Again, one can observe that GNN-SD achieves consistent and considerable performance enhancement against all the baselines. On classification of PROTEINS, for example, even the next to last model trained with GNN-SD (GraphSage, 76.71%) outperforms the best model (GAT, 76.36%) trained in the ordinary way. The results further validate the generalization ability of our self-distilling strategy.

Dataset	Model	Baseline	w/ GNN-SD	Gain
ENZYMES	GCN	$64.00_{\pm 5.63}$	$66.66_{\pm 3.94}$	(+2.66)
	GAT	$65.33_{\pm 5.90}$	$68.00_{\pm 2.66}$	(+2.66)
	GraphSage	$68.33_{\pm 6.41}$	$\mathbf{70.00}_{\pm 5.05}$	(+1.66)
	GIN	$66.00_{\pm 7.19}$	$\mathbf{69.33}_{\pm 4.02}$	(+3.33)
	GatedGCN	$65.33_{\pm 4.52}$	$67.33_{\pm 1.33}$	(+2.00)
DD	GCN	$77.83_{\pm 1.02}$	$\mathbf{78.67}_{\pm 1.68}$	(+0.84)
	GAT	$76.65_{\pm 2.51}$	$77.50_{\pm 2.50}$	(+0.85)
	GraphSage	$76.14_{\pm 1.66}$	$77.49_{\pm 1.89}$	(+1.35)
	GIN	$73.00_{\pm 3.90}$	$74.77_{\pm 3.50}$	(+1.77)
	GatedGCN	$77.83_{\pm 1.67}$	$\mathbf{78.20}_{\pm 1.98}$	(+0.37)
PROTEINS	GCN	$75.55_{\pm 2.91}$	$76.81_{\pm 3.19}$	(+1.26)
	GAT	$76.36_{\pm 2.77}$	$\mathbf{77.53}_{\pm 3.38}$	(+1.17)
	GraphSage	$75.55_{\pm 4.02}$	$76.71_{\pm 3.81}$	(+1.15)
	GIN	$64.86_{\pm 3.03}$	$70.06_{\pm 4.89}$	(+5.20)
	GatedGCN	$76.36_{\pm 3.94}$	$\mathbf{76.90}_{\pm 3.68}$	(+0.54)

Table 3: Graph classification on various GNN backbones.

Compatibility Evaluation There exist other methods that aim at facilitating the training of GNNs. One influential work is DropEdge, which randomly samples graph edges to introduce data augmentation. We conduct experiments on JKNet and Citeseer dataset to evaluate the compatibility of these orthogonal training schemes. The edge sampling ratio in DropEdge is searched at the range of $\{0.1, ..., 0.9\}$, with results demonstrated in Table 4. It reads that while both GNN-SD and DropEdge are capable of improving the training, GNN-SD might perform better on deep backbones. Notably, employing them concurrently is likely to deliver further promising enhancement.

Models	Layers			
	8	16	32	50
Baseline	$78.1_{\pm 0.9}$	$79.1_{\pm 1.1}$	$79.3_{\pm 0.8}$	$78.4_{\pm 1.1}$
w/ DropEdge	$79.4_{\pm 0.7}$	$79.3_{\pm 0.9}$	$79.4_{\pm 1.2}$	$78.7_{\pm 0.9}$
w/ GNN-SD	$79.1_{\pm 0.8}$	$\mathbf{79.9}_{\pm 0.5}$	$79.7_{\pm 1.1}$	$79.2_{\pm 0.8}$
w/ Both	$\mathbf{80.1}_{\pm 0.6}$	$79.6_{\pm 0.8}$	$\mathbf{80.2}_{\pm 0.8}$	$\mathbf{79.6}_{\pm 0.5}$

Table 4: Compatibility study. Experiments are conducted under the full-supervised scheme, following DropEdge's implementation.

	GNN-*B*	GNN-*L*	GNN-*N*	GNN-*G*	GNN-*M*	
$\mathcal{L}_{\mathrm{L}}$		√			√	√
$\mathcal{L}_{\mathrm{N}}$			√		√	√
$\mathcal{L}_{\mathrm{G}}$				√		√
Cora	80.12	79.83	80.83	/	**81.16**	/
Citeseer	70.53	71.16	71.43	/	**71.52**	/
Pubmed	78.52	78.53	**79.44**	/	79.26	/
ENZYMES	66.00	66.66	67.66	67.66	66.66	**69.33**
DD	73.00	73.76	**74.77**	73.51	73.76	74.14

Table 5: Ablation study of different knowledge sources.

Ablation Studies We perform an ablation study to evaluate the knowledge sources and identify the effectiveness of our core technique, as shown in Table 5. We select GAT as the evaluation backbone for node classification and GIN for graph classification. We name the baseline as 'GNN-*B*', and model solely distilled by intermediate logits [2019], neighborhood discrepancy, and compact graph embedding as 'GNN-*L*', 'GNN-*N*', 'GNN-*G*', respectively. The models distilled by mixed knowledge are represented as 'GNN-*M*'. One observation from the results is that simply adopting the intermediate logits seems to fail in bring consistent improvement (highlighted in gray), while it may cooperate well with other sources since it promotes the updating of shallow features (highlighted in blue). In contrast, the discrepancy retaining plays the most important role in distillation training. For graph classifications, the involvement of compact graph embedding also contributes well while jointly works with the others.

5 Conclusion

We have presented an efficient self-distillation framework tailored for GNNs. Serving as a drop-in replacement of the standard training process, it yields consistent and considerable enhancement on various GNN models.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (61972219), the RD Program of Shenzhen (JCYJ20190813174403598, SGDX20190918101201696), the National Key Research and Development Program of China (2018YFB1800601).

References

[Ahn *et al.*, 2019] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[Bresson and Laurent, 2017] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[Chen *et al.*, 2018] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

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

[Chen *et al.*, 2020b] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. *arXiv preprint arXiv:2007.02133*, 2020.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[Heo *et al.*, 2019] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. *arXiv preprint arXiv:1904.01866*, 2019.

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

[Hou *et al.*, 2019] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1013–1021, 2019.

[Huang *et al.*, 2018] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. *Advances in Neural Information Processing Systems*, 31:4558–4567, 2018.

[Kersting *et al.*, 2016] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016.

[Kim *et al.*, 2018] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NeurIPS*, 2018.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606*, 2018.

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

[Romero *et al.*, 2014] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[Rong *et al.*, 2019] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2019.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[Tian *et al.*, 2019] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2019.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Xu *et al.*, 2018a] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2018.

[Xu *et al.*, 2018b] 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*, 2018.

[Yang *et al.*, 2020] 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*, pages 7074–7083, 2020.

[Yuan *et al.*, 2020] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[Zhang *et al.*, 2019] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[Zitnik and Leskovec, 2017] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.