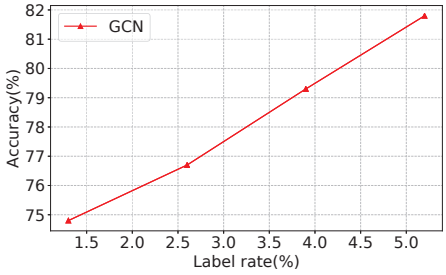


Figure 1: Accuracy of node classification on Cora with different label rates.

Still, it seems that GCN fails to make full use of the potential inside the unlabeled data. GCN only captures the already available information about the unlabeled data, e.g., their features and their adjacent relations, which makes the embedding of the unlabeled data highly dependent on the adjacent labeled data. With limited labels, GCN fails to propagate them to the entire graph, decreasing the information buried inside a large portion of unlabeled data. Figure 1 displays the accuracy of a regular GCN obtained on the Cora citation network [33] when the label rate ranges from 1.3% to 5.2%. The performance of GCN drops quickly as the labeled training size shrinks. To reduce the dependency on labeled data, we can select and label some unlabeled data and add them to the training set. Their embeddings will be independently updated during the back-propagation [43], instead of depending on their adjacent labeled data alone.

Generating pseudo labels for the unlabeled data is a widely used idea in SSL and the most representative methods are the Co-Training and Self-Training [14]. Co-Training relies on a random walk [56] to complement the GCN by exploring the global graph topology [35]. Self-Training first generates labels for unlabeled data and then selects the most confident predictions for each class. The selected new labeled data is added to the training set and then helps to train a more powerful model. Still, both of the methods suffer from the same shortcoming: the learned labels may not be correct. It is hard to determine whether the trained models have correctly predicted those labels. In addition, not only the learned labels contain less information than the probability distribution of the class, e.g., the softmax probability, but the threshold used to generate the label is hard to tune.

To overcome these issues while producing pseudo labels, the Consistency Regularization [52] method has been proposed recently. Unlike Self-Training and Co-Training, Consistency Regularization assumes that the prediction of similar inputs should be similar, even if subjected to slight interference. Consistency Regularization does not generate pseudo labels but adds a penalty term related to unlabeled data to the loss function. Among all Consistency Regularization methods, the teacher-student consistency based on Knowledge

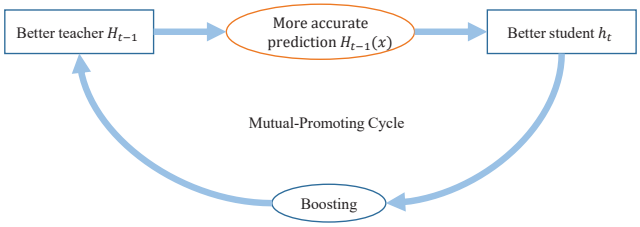


Figure 2: The training pipeline of RDD

Distillation (KD) [25] has been widely used due to its superior performance. In KD, a teacher model is first learned and then its predictions on unlabeled data are used to learn a student model. Compared with the predicted label given by Self-Training and Co-Training, the generated node embeddings by the teacher model contain much more information, resulting in a lower variance in the gradient between training cases [25]. The student model mimics the node embeddings predicted by the teacher model. The better the teacher model is, the more accurate its predictions on unlabeled data will be, resulting in better learning for the student model.

By combining the predictions of several models, Ensemble Learning [57] has been widely used in KD. The combination of several student models significantly improves the performance [21, 46]. For example, the Mean Teacher [46] trains each student model under the consistency regularization of a teacher model and constantly improves the performances of the teacher by weighted averaging. Similarly, the student model in Born Again Neural Networks (BANs) [21] is also under the supervision of the teacher model. Based on Ensemble learning, BANs combines all the pre-trained models to construct a more powerful ensemble model.

Despite the important improvement the KD techniques provide, their performances are considerably linked to the ones of the teacher. If a data point is wrongly classified by the teacher, all the students will learn this wrong label. Taking all the predictions of the teacher model as ground truth will result in a high bias of the student model. Besides, the student models mimic all the teacher outputs in KD and then they are similar to each other, which is called limited diversity [60]. Because of the limited diversity and the high bias, KD-based ensemble methods only provide a limited gain.

In this paper, we introduce *reliability* as a way to improve KD: reliable data –be it a reliable node or a reliable edge – will be used during the training while the *unreliable* data will be discarded. Our resulting approach, dubbed *Reliable Data Distillation (RDD)*, is a data driven semi-supervised Graph Convolutional Network training method which is built on top of KD using reliability.

RDD not only cares about data quantity but also considers the individual quality, i.e. *the node reliability and edge reliability*, of the graph data. Unlike regular KD methods, the

student model in RDD does not mimic the teacher model predictions. Instead, it exploits the reliability of those predictions to improve its learning. The student model first assesses whether the teacher's predictions are reliable, and then uses the reliability to improve its learning. Compared with KD, the student model is more likely to predict the correct labels on the unreliable data and the diversity between teacher and student can be enhanced accordingly. After the training of the student model, the teacher model is updated using the student model, so the teacher model can benefit from the learning of the student model. From an Ensemble Learning point of view, our approach is a Self-Boosting method as shown in Figure 2. We constantly generate new student GCN models to improve the performances of the teacher model and make the predictions of both teacher and student more and more accurate at the same time.

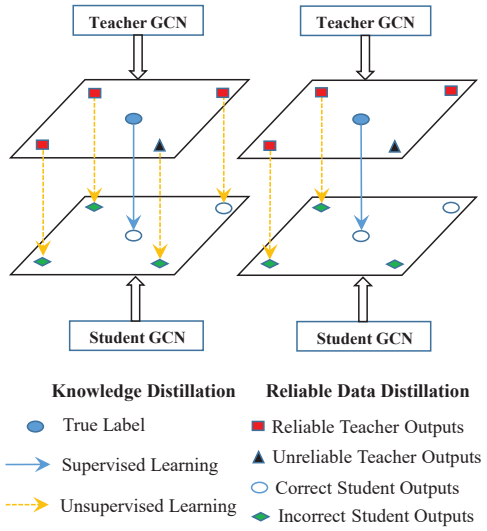


Figure 3: Student learning for both Knowledge Distillation and Reliable Data Distillation.

1.2 Overview of Technical Contributions

In the following, we introduce the key parts of our proposed method and then describe each contribution individually.

Reliable Node Distillation. As discussed in Section 1.1, the teacher-student consistency based on KD has not considered the reliability of predicted node embeddings. This requires unnecessary training budget and causes high bias of student model since the student model repetitively relearns all the data knowledge predicted by the teacher model. We propose Node Reliability to solve these problems. During each training epoch in RDD, the student evaluates the reliability of the node embeddings predicted by the teacher. For the labeled data, we only consider whether these data have been correctly classified. For the unlabeled data, we consider Information Entropy [6] of their predicted softmax outputs. Indeed softmax output with high information entropy means the classifier is not sure about its prediction, therefore the prediction can be seen as unreliable.

As shown in Figure 3, the student mimics all the outputs of the teacher without selection, resulting in learning with wrong labels from unreliable prediction from the teacher model. RDD tackles this issue by first classifying the predictions of the teacher model in two classes: reliable or unreliable. The reliable outputs are then used to correct the student model's wrong predictions. Note that here correct/incorrect refers to the prediction of the teacher model and not the true labels (which are unknown). In the example, in addition to the labeled data, the student will learn the reliable knowledge it wrongly predicts compared to the teacher. Besides, unreliable predictions are not taught to the student models and neither are the reliable predictions made by the teacher models that are correctly predicted by the student model.

Reliable Edge Distillation. We use Graph Laplacian Regularization [33] to jointly model graph structures and node features. With an explicit graph-based regularization [29], the label information can be smoothed over the graph and the performances of GCN can be enhanced accordingly. However, Graph Laplacian Regularization assumes that two nodes with an edge connection are more likely to have the same class and thus force adjacent nodes to have similar node embeddings. This simple heuristic assumption fails to grasp the more complex relationships of many nodes in the graph, especially for nodes lying near the decision boundary. Those nodes are actually the ones on which predictions are unreliable. We introduce edge reliability as a way to adapt the regularization and regularize only reliable edges.

Ensemble Learning for graph data. Current existing KD-based ensemble methods are not designed specifically for GCN. Let aside the reliability, they do not consider the structure and locality of graphs. For example, highly connected nodes, i.e. with a large number of edges, influence many nodes and thereby play a more important role during the training of a GCN than less connected nodes. RDD uses PageRank [41] to measure the importance of each node in the final ensemble. Furthermore, diversity among the base models in an ensemble system provides an important increase in the final accuracy of the system [57]. In existing KD-based methods, the student model mimics the teacher models, resulting in reduced diversity. However, in RDD, the student model actively learns the reliable data knowledge predicted by the teacher, so it has more chance to relearn the unreliable data and make different predictions on these data. In this way, the diversity problem can be solved.

Self-Boosting SSL Framework. We introduce a new Self-Boosting training framework to improve the accuracy of

the knowledge predicted by the teacher. After training each student model, we form an ensemble with the former pre-trained student models and set this ensemble as the teacher model for the next training epoch. At each training iteration, the teacher model is improved and predicts more accurate labels to be learned by the student. Conversely, under more accurate supervision, the training of the student model is improved and the model becomes more and more accurate. As a result, both the teacher and student models benefit from this Self-Boosting cycle.

Contributions The main contributions of our work can be summarized as follows: *(1)* To the best of our knowledge, we are the first to **improve Knowledge Distillation** by emphasizing the data reliability. *(2)* We propose a RDD framework to capture the **reliability** in both nodes and edges for GCN. *(3)* We propose **a new Self-Boosting SSL framework** as a graph-driven ensemble method for GCN. *(4)* We have conducted an **extensive evaluation** of our approach on real-world datasets, the results have demonstrated its superior performances over state-of-the-art approaches.

2 PRELIMINARIES

2.1 Definition of notations

To help the readers understand this work, we begin by introducing some notations related to RDD in Table 1.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$, an adjacency matrix $A \in \mathbb{R}^{N\times N}$. The node feature information matrix is represented as $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}..., \boldsymbol{x_N}\}$ and $\boldsymbol{x_i} \in \mathbb{R}^d$. The node labels are represented as one-hot vector $Y = \{\boldsymbol{y_1}, \boldsymbol{y_2}..., \boldsymbol{y_N}\}$ and $\boldsymbol{y_i} \in \mathbb{R}^k$. The node set is partitioned into labeled node set $\mathcal{V}_\mathrm{l}$ and unlabeled node set $\mathcal{V}_\mathrm{u}$. The goal in semi-supervised learning is to predict the labels of the unlabeled nodes $\mathcal{V}_\mathrm{u}$.

2.2 Graph convolutional network

A multi-layer GCN follows the layer-wise propagation rule. At layer l, the output is the hidden representation $H^{(l)}$:

$$H^{(l)} = \delta\big(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l)}\big), \tag{1}$$

where $\widetilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_N is the identity matrix, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\delta(\cdot)$ denotes an activation function, such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $H^{(l-1)}$ is the input of lth layer and $H^{(0)} = X$. For a L-layer GCN on semi-supervised node classification on a graph with a symmetric adjacency matrix A, we first compute $\widetilde{A}$ in a pre-processing step and the forward model takes the simple form:

Table 1: NOTATIONS

Symbols	Definitions
v_i	The ith node
$\boldsymbol{x_i}$	The features of the ith node
$\boldsymbol{y_i}$	The one hot encoding of the ith label
k	The class of data
N	The size of the training set
T	The number of training iterations
I	The information entropy
γ	Controls the proportion of knowledge transfer
β	Controls the strength of the edge regularization
p	Controls the threshold of node reliability
h_t	The tth student model
H_t	The tth teacher model
α_t	The weight of the tth base model
$h_t(\boldsymbol{x_i})$	The tth base model's label prediction of v_i
$w_{i,j}$	The edge reliability between v_i and v_j
$\boldsymbol{h_t(x_i)}$	h_t's softmax outputs on v_i
$\boldsymbol{H_t(x_i)}$	H_t's softmax outputs on v_i
$\boldsymbol{F_t(x_i)}$	H_t's predicted node embedding on v_i
$\boldsymbol{f_t(x_i)}$	h_t's predicted node embedding on v_i
$Pr(\boldsymbol{x_i})$	The pagerank value of v_i

$$\begin{aligned} H^{(l)} &= \mathrm{ReLU}\big(\widetilde{A}H^{(l-1)}W^{(l)}\big), l = 1..., L, \\ Z &= \mathrm{softmax}\big(H^{(L)}\big). \end{aligned} \tag{2}$$

$H^{(l)} \in \mathbb{R}^{N\times D^{(l)}}$, where $D^{(l)}$ is the parameter for the embeddings dimension for the lth layer, and $D^{(L)} = k$. The softmax activation function, defined as $\mathrm{softmax}(x_i) = \frac{1}{Z}\exp(x_i)$ with $Z = \sum_i \exp(x_i)$, is applied row-wise. We then evaluate the cross-entropy error over all labeled examples:

$$\mathcal{L} = -\sum_{v_i \in \mathcal{V}_\mathrm{l}} y_i \log Z_i, \tag{3}$$

Equation 1 shows that the information contained in a node is propagated to its neighbors at each layer: a K-layer GCN only propagates node information up to K-hops neighborhood. When only a few labels are given, e.g., a 5% label rate, a shallow GCN cannot effectively propagate the labels to the entire data graph. For a shallow GCN, the unlabeled nodes may get limited attention and make less contribution to the learning process, thus the performance of GCN drops quickly as the labeled training size shrinks.

A natural approach to make better use of the unlabeled data is to use a deeper network [2, 51]. For example, the residual connection [33] enables GCN to carry over information from the previous layer's input. Even with residual connections, GCNs with more layers do not perform as well as the 2-layer GCN on many datasets, e.g., the citation networks. Indeed, stacking more layers into a GCN leads to the *over-smoothing problem* [35], which means back-propagation through the network eventually leads to the convergence of the features of nodes to the same value. In fact, most state-of-the-art GCNs are no deeper than 4 layers, which limits the representation ability of GCN [35].

Existing GCNs and their variants cannot make full use of the unlabeled nodes. We propose RDD to solve this problem by considering a data-oriented approach. The teacher model in RDD can generate reliable node embeddings for unlabeled nodes, and these nodes can be trained with back-propagation, not exclusively by using the label information contained in their adjacent nodes. As a result, the unlabeled nodes can be used no matter how far they are from labeled nodes.

2.3 Ensemble Learning

Ensemble Learning [13] is a class of machine learning techniques that combines the predictions of different models into a more accurate one. Ensemble Learning has demonstrated its effectiveness in many problems in which it has become a part of the most efficient state-of-the-art techniques, such as in the notorious Netflix Prize [4] and various Kaggle competitions [26]. It is also widely used in semi-supervised tasks, as the knowledge learned on the unlabeled data can be used to train a new classifier that is then combined to the previous ones to obtain an even more accurate classifier.

One of the most widely used ensemble methods is Bagging [8]. Bagging trains multiple models on sampled training sets and then combined them into one. The final model predicts the class of an item by making each base model estimating the class of the item and then selecting the most plausible class out of these estimations. While Bagging aims at reducing the variance of the model's predictions, Boosting [20, 30] aims at lowering the bias of the model. To achieve this, Boosting combines multiple *weaker* models into a single strong one [31, 59]. Similarly, Stacking [9] combines the outputs of several base models by feeding them to another algorithm that combines them to make the final predictions.

Based on these three methods, many works have been proposed in recent years. For example, XGBoost is an efficient implementation of gradient Boosting decision trees [17]. A neural network in Snapshot Ensemble [27] converges to different local minimums along its optimization path. Unlike Bagging [9], which trains each network independently, Snapshot Ensemble saves the model parameters before resetting the learning rate and treats each model replica as a base model. Besides, Born Again Neural Networks [21] initializes each base model from a different random seed and each of them is trained from the supervision of the earlier model.

Evidence suggests that the two key factors for the efficient use of Ensemble Learning techniques are the diversity and the accuracy of the base models used [13]. However, from diversity and accuracy point-of-view, the above methods are not suitable for GCN. Bagging-based methods individually train each base model on a subsampled training set, leading to a low accuracy for each base model, especially only a few labeled nodes are available. Furthermore, semi-supervised techniques usually overfit the labeled nodes in GCN [33], making Boosting unsuitable since the base models can almost classify all the labeled nodes. Finally, both BANs and Snapshot Ensemble face the problem of limited diversity since they train each base model based on the former one.

Considering the accuracy and diversity of base models, all the existing ensemble methods are not suitable for learning over GCN. According to the characteristic of GCN and graph data, we introduce the concept of data reliability and propose a new graph-based ensemble method in RDD. Based on the reliability, the bias of base models is reduced since the student cannot only be trained with the label information like the traditional GCN but also get reliable supervision from the teacher model and Graph Laplacian Regularization. Besides, the diversity in RDD is higher than BANs because the student model in RDD actively learns the reliable data knowledge predicted by the teacher, so it has more chance to relearn the unreliable data and make different predictions on these data.

2.4 Knowledge Distillation

Knowledge Distillation (KD) [25] is proposed to transfer the knowledge acquired by one model (the teacher) to another model (the student) that is typically smaller. In particular, KD is widely used in model compression [11]. Knowledge Distillation aims at minimizing the following loss function:

$$\mathcal{L}_{KD} = \sum_{i=1}^{N} l\big(h(x_i), H(x_i)\big), \tag{4}$$

where l is a loss function that measures the prediction distance between the teacher model H and student model h. Besides the supervision of the labeled data, h also gets extra supervision from a powerful teacher model H. By doing so, it can achieve higher accuracy in many classification tasks than a regular model trained only on the labeled data.

KD can achieve more than just improving the training of a student model: recent works have combined KD with Ensemble Learning to improve even further the accuracy of the final model. BANs [21] and Mean Teacher [46] interactively integrate the distilled student models into an ensemble

model. The resulting ensemble model is much more accurate than a single model.

Unfortunately, all these KD-based ensemble methods do not consider the reliability of the teacher's predictions, leading to inaccurate base models and limited diversity. To better use the knowledge from the teacher model, we improve KD by introducing the concept of data reliability.

3 RELIABILITY IN GRAPH DATA

In classical KD, the student models consider the outputs of the teacher model as ground truth, even the misclassified ones. Not only it provokes a bias, but the fact that the students learn all predicted knowledge from the teacher lowers the diversity in the ensemble. We introduce the notion of **reliability** to address these two issues. Reliability answers the following question: *can I trust this data for my learning*? Data reliability in RDD is used to make a distinction between correct and incorrect teacher outputs, to prevent the student to learn the incorrect knowledge. For graph data, the reliability of the data also determines what nodes and edges should be taken into account when propagating information during the training of GCN.

3.1 Node reliability

In order to avoid the student models to mimic the wrong teacher outputs, we introduce the **node reliability**. The node reliability is a property of the nodes stating whether their outputs by the teacher model can be used for training the student model. By preventing the student models to learn the mistakes of the teacher models, node reliability reduces the bias of the overall model.

For labeled nodes, knowing if the output can be used is easy: if the output is the same as the label, the node is reliable, otherwise the node is unreliable. For unlabeled nodes, however, assessing the correctness of the output is difficult since we do not know their labels. We use the concept of Information Entropy to evaluate if the probability that the output is correct: the lower the entropy, the higher the certainty in the prediction [32]. Instead of using a threshold that may vary significantly for different data and models, we consider that the p-percent outputs with the lowest entropy are seen as correct while all the others are considered as incorrect. Furthermore, from an Ensemble Learning perspective, the prediction is more certain if all the base models predict the same labels for a given node. To ensure this, we only consider the nodes with a correct output for which the student and the teacher models predict the same labels.

To summarize, a **node is reliable** if *(1)* it is a labeled node and the predicted label of the teacher is correct or *(2)* it is an unlabeled node, the entropy of its prediction by the teacher model is among the p-percent lowest, and its predicted labels by the teacher and student models are the same. **All the other nodes are unreliable**.

Algorithm 1 Update the node reliability

Require:
Dataset: X and Y;
E: Number of training epochs;
H: The teacher model;
Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
Ensure:
$\mathcal{V}_\mathrm{r}$: Reliable nodes
$\mathcal{V}_\mathrm{b}$: Nodes teacher learns reliably but student learns incorrectly
1: $\forall \boldsymbol{x_i}, I_H(\boldsymbol{x_i}) = -\boldsymbol{H(x_i)} \log \boldsymbol{H(x_i)}$
2: sort $I_H(\boldsymbol{x})$ in ascending order
3: **for** $e = 1$ to E **do**
4: $\quad \forall v_i \in \mathcal{V}_\mathrm{l}$, add v_i to $\mathcal{V}_\mathrm{r}$ if $h_e(x_i) = y_i$
5: $\quad \forall \boldsymbol{x_i}, I_{h_e}(\boldsymbol{x_i}) = -\boldsymbol{h_e(x_i)} \log \boldsymbol{h_e(x_i)}$
6: $\quad$ sort $I_{h_e}(\boldsymbol{x})$ in descending order
7: $\quad \forall v_i \in \mathcal{V}_\mathrm{u}$, add v_i to $\mathcal{V}_\mathrm{r}$ if $I_H(\boldsymbol{x_i}) \in$ top p% of I_H
8: $\quad \forall v_i \in \mathcal{V}_\mathrm{r}$, remove v_i from $\mathcal{V}_\mathrm{r}$ if $h_e(x_i) \neq H(x_i)$
9: $\quad \forall v_i \in \mathcal{V}_\mathrm{r}$, add v_i to $\mathcal{V}_\mathrm{b}$ if $I_{h_e}(\boldsymbol{x_i}) \in$ top p% of I_{h_e}
10: **end for**

Algorithm 1 shows how the node reliability is updated during each training epoch. In Algorithm 1, we first compute the information entropy $I_H(\boldsymbol{x})$ for every node feature, using the predicted softmax output $\boldsymbol{H(x_i)}$ made by the teacher model (line 1 in Algorithm 1). Next, these values are sorted in ascending order (line 2 in Algorithm 1). Labeled nodes are seen as reliable if they have been correctly classified (line 4 in Algorithm 1). For unlabeled nodes, the information entropy $I_{h_e}(\boldsymbol{x})$ is computed using the student model's predictions. The nodes are considered as reliable if they have consistent label predictions with the former teacher model H_T and if the predicted information entropy $I_H(\boldsymbol{x_i})$ is low in the top p% of $I_H(\boldsymbol{x})$ (line 7-8 in Algorithm 1). Unlike $I_H(\boldsymbol{x})$, the values of $I_{h_e}(\boldsymbol{x})$ are sorted in decreasing order (line 6 in Algorithm 1). For each reliable node v_i, if $I_{h_e}(\boldsymbol{x_i})$ is high in top p% of the sorted I_{h_e} (line 9 in Algorithm 1), the student I_{h_e} is unsure of its prediction $h_e(x_i)$. This means the student learns data v_i incorrectly but teacher learns it reliably, so we add it to the set $\mathcal{V}_\mathrm{b}$ (line 9 in Algorithm 1).

3.2 Edge reliability

Existing GCNs generally fail to consider the local consistency in the training process. That is, if the connected two nodes share similar features, then their labels and representations should also be similar. A simple method [47] to solve this problem is Graph Laplacian Regularization [33] (GLR). GLR relies on the heuristic that two nodes sharing an edge connection and similar features are more likely to have the same class and should have similar node embedding. As a

Algorithm 2 Update the edge reliability

Require:
Dataset: X and Y;
E: Number of training epochs;
H: The teacher model;
A: The adjacent matrix;
Graph $\mathcal{G} = (\mathcal{V}_{\mathrm{r}}, \mathcal{E})$;
Ensure:
The reliable edge:$\mathcal{E}_{\mathrm{r}}$
1: Initialize B and C with the Zero Matrix
2: **for** $e = 1$ to E **do**
3: $\forall(v_i, v_j) \in \mathcal{E}$, set $B_{i,j} = 1$ if v_i and v_j are both in $\mathcal{V}_{\mathrm{r}}$
4: $\forall(v_i, v_j) \in \mathcal{E}$, set $C_{i,j} = 1$ if $h_e(x_i) = h_e(x_j)$
5: $w = A * B * C$
6: $\forall(v_i, v_j) \in \mathcal{E}$, add (v_i, v_j) to $\mathcal{E}_{\mathrm{r}}$ if $w_{i,j} = 1$
7: **end for**

consequence, GLR forces adjacent nodes to have similar embeddings. While this assumption seems intuitive, it fails to grasp the complexity of many real cases such as an edge between two nodes with different classes. We introduce **edge reliability** to refine the heuristic used during GLR.

While it is true that two connected nodes have similar node embedding if they have the same predicted label [39], the challenge is to be sure that their predicted labels are correct. As seen in Sec. 3.1, using node reliability prevents from using the teacher's predictions that may be wrong. Two nodes should have similar embedding only if they are both reliable and their predicted labels are the same.

An edge is **reliable** if both its nodes are reliable and they have the same predicted class. Algorithm 2 shows our edge reliability-computation. After each iteration, we use $w_{i,j}$ to measure the edge reliability between two node embeddings $\boldsymbol{x_i}$ and x_j (line 5 in Algorithm 2). Suppose the immediate neighborhood of $\boldsymbol{x_i}$ is $\{x_j, x_j \in N(\boldsymbol{x_i})\}$, we have

$$w_{i,j} = A_{i,j} * B_{i,j} * C_{i,j}, \tag{5}$$

where A, B and C are all zero matrix. We have $A_{i,j} = 1$, if the two nodes are linked. $B_{i,j} = 1$, if the two nodes are reliable (line 3 in Algorithm 2). $C_{i,j} = 1$, if their predictions $h_t(\boldsymbol{x_i})$ and $h_t(x_j)$ belong to the same class (line 4 in Algorithm 2). After the updating of the matrix w, we considered the edge (v_i, v_j) as reliable if $w_{i,j} = 1$ (line 6 in Algorithm 2).

4 RELIABLE DATA DISTILLATION

4.1 Overview

Our Reliable Data Distillation method follows the KD framework with two roles: teacher model and student model. The teacher model is an ensemble model, it is a combination of all the previous student models. It aims at producing reliable predictions for the unlabeled nodes. The student model is a basic GCN model and it is trained over the reliable nodes,

Algorithm 3 Reliable Data Distillation

Require:
Dataset: X and Y;
T: Number of base GCNs;
β: Controls the proportion of knowledge transfer;
γ: Controls the strength of edge regularization;
Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
Ensure:
The boosted classifier:H_T
1: $t = 1$;
2: $h_1 \leftarrow \mathrm{GCN}(\mathcal{V}, \mathcal{E})$
3: $\forall \boldsymbol{x_i}, I_1(\boldsymbol{x_i}) = -\boldsymbol{h_1}(\boldsymbol{x_i}) \log \boldsymbol{h_1}(\boldsymbol{x_i})$
4: $\alpha_1 = \frac{1}{\sum_{i=1}^{N} I_1(\boldsymbol{x_i}) Pr(x_i)}$
5: Update H_1 with h_1 and α_1
6: **for** $t = 2$ to T **do**
7: Update $\mathcal{V}_{\mathrm{r}}$ and $\mathcal{E}_{\mathrm{r}}$ in each epoch
8: **for** v_i in $\mathcal{V}_{\mathrm{l}}$ **do**
9: $\mathcal{L}_1 += -\boldsymbol{y_i} \log \boldsymbol{h_t}(\boldsymbol{x_i})$
10: **end for**
11: **for** v_i in $\mathcal{V}_{\mathrm{b}}$ **do**
12: $\mathcal{L}_2 += \|\boldsymbol{f_t}(\boldsymbol{x_i}) - \boldsymbol{F_{t-1}}(\boldsymbol{x_i})\|^2$
13: **end for**
14: **for** (v_i, v_j) in $\mathcal{E}_{\mathrm{r}}$ **do**
15: $\mathcal{L}_{\mathrm{reg}} += \left\|\boldsymbol{f_t}(\boldsymbol{x_i}) - \boldsymbol{f_t}(\boldsymbol{x_j})\right\|^2$
16: **end for**
17: $\mathcal{L} = \mathcal{L}_1 + \gamma \mathcal{L}_2 + \beta \mathcal{L}_{\mathrm{reg}}$
18: $h_t \leftarrow \mathrm{GCN}(\mathcal{V}, \mathcal{E}, H_{t-1}, L)$
19: $\forall \boldsymbol{x_i}, I_t(\boldsymbol{x_i}) = \boldsymbol{h_t}(\boldsymbol{x_i}) \log \boldsymbol{h_t}(\boldsymbol{x_i})$
20: $\alpha_t = \frac{1}{\sum_{i=1}^{N} I_t(\boldsymbol{x_i}) Pr_t(\boldsymbol{x_i})}$
21: Update H_t with h_t and α_t
22: **end for**
23: $H_T = \sum_{i=1}^{T} \alpha_t h_t$

including both the labeled nodes and the unlabeled nodes with the node embeddings predicted by the teacher model.

Figure 4 displays our RDD framework with a two-layers GCN. The teacher model in RDD firstly predicts the last layer's node embedding. Secondly, the student measures the data reliability in each training epoch based on the outputs of the teacher and student. After getting reliable data, the student will be trained with two types of data: *(i)* the labeled data and *(ii)* the reliable data the student model misclassified. Finally, we integrate the student into the ensemble system to construct a more powerful teacher.

The training procedures of RDD. Algorithm 3 shows the pseudo-code of our training procedure. A regular GCN is trained on the training set as the first student model (line 2 in Algorithm 3) and the initial teacher model is generated from it (line 3-5 in Algorithm 3). The teacher model predicts outputs for all nodes. The reliability of the teacher outputs is measured and the outputs are then classified into reliable and unreliable outputs (line 7 in Algorithm 3). Each GCN

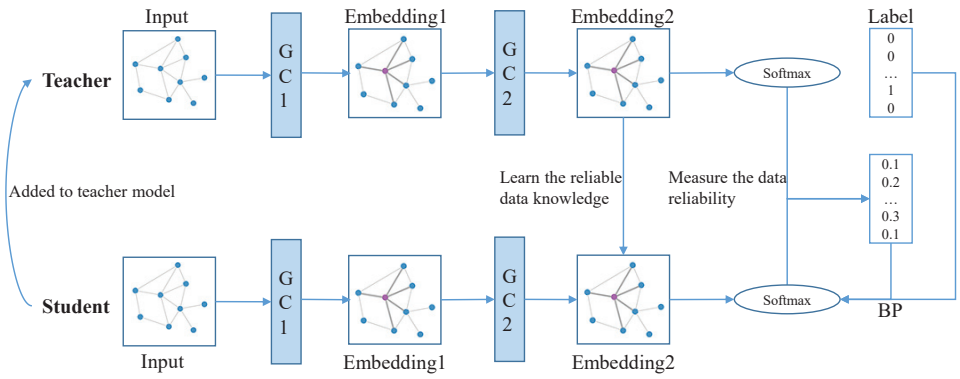


Figure 4: Overview of Reliable Data Distillation

student model is trained under the supervision of the reliable knowledge given by the teacher. Once trained, the student model is integrated into the ensemble system to get a more powerful teacher model (line 18-21). We propose a new loss function for the training of the student, which contains the supervised loss $\mathcal{L}_1$ (line 8-10 in Algorithm 3), the unsupervised loss $\mathcal{L}_2$ (line 11-13 in Algorithm 3) and the regularization loss (line 14-16 in Algorithm 3). The training process is iteratively done T times, and we combine T base models with the ensemble.

In the following, we describe more in details two keys components of our approach: the Reliable Data Driven GCN (Sec. 4.2) that makes the student model learns from reliable data; and the Graph Data Based Ensemble (Sec. 4.3) that considers data reliability and data importance.

4.2 Reliable Data Driven GCN

We now present how we make the full use of the reliability of the data to improve semi-supervised KD GCN. Node and edge reliability are used to provide Reliable Node Distillation and Reliable Edge Distillation respectively.

4.2.1 ***Reliable Node Distillation.*** The training process of reliable node distillation is shown in Figure 5. The teacher model produces the reliable node set while the student model produces the incorrect node set. The incorrect nodes here refer to nodes whose prediction's entropy by *the student's model* is among the p-percent highest. What is incorrect is not necessarily the label, but the corresponding embeddings which have high entropy. A new loss function is used to learn over the incorrect predicted data by the student model.

For each node $v_i \in \mathcal{V}_b$, the teacher model H_{t-1} has learned it reliably and the current student model h_t has leaned it incorrectly. Usually, the teacher model is more powerful than the current student model h_t. To correct what h_t misclassified, the student model learns its correct labels from H_{t-1}.

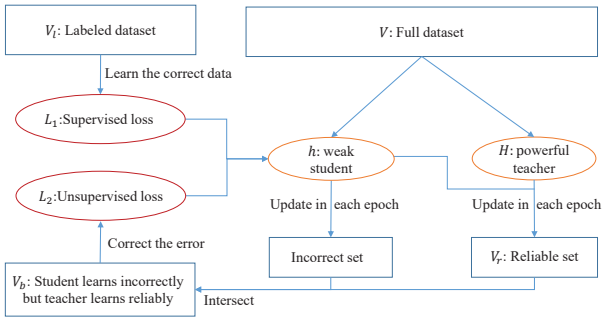


Figure 5: Training process of reliable node distillation

Like the traditional GCN, we firstly apply the supervised loss on the labeled nodes as shown in Eq. 6.

$$\mathcal{L}_1 = -\sum_i \boldsymbol{y_i} \log \boldsymbol{h_t}(\boldsymbol{x_i}), \forall v_i \in \mathcal{V}_l, \tag{6}$$

To correct its wrong values, the student model h_t tries to mimic the embedding $\boldsymbol{F_t}(\boldsymbol{x_i})$ of each reliable node $\boldsymbol{x_i}$. Similarly to KD, we use a student model h_t to mimic the outputs of the teacher model H_{t-1}. However, we have two main differences. First, we mimic the whole node embeddings, not the softmax outputs which are used in KD, since the original node embedding contains more information than the softmax outputs. Besides, the student model h_{t-1} actively learns reliable knowledge from the teacher model while traditional KD just learns all the knowledge without selection. We formulate our method as a part of loss function:

$$\mathcal{L}_2 = \|\boldsymbol{f_t}(\boldsymbol{x_i}) - \boldsymbol{F_{t-1}}(\boldsymbol{x_i})\|^2, \forall v_i \in \mathcal{V}_b. \tag{7}$$

For the training pipeline, we train H_{t-1} to correct the error of the student model h_t.

4.2.2 ***Reliable Edge Distillation.*** We briefly review the basic assumptions of graph representation learning study:

- **Assumption 1.** Two nodes with edge connection are more likely to have the same class.
- **Assumption 2.** Two nodes with edge connection have similar node embeddings.

Based on Assumption 1 and 2, the semi-supervised graph node classification problem can be framed as a Graph Laplacian Regularization [33]:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_{\text{reg}}, \text{with } \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{i,j} \|f(\boldsymbol{x_i}) - f(x_j)\|^2, \quad (8)$$

where $\mathcal{L}_1$ denotes the supervised loss and $f(\cdot)$ is the label map function. The formulation of Eq.8 relies on Assumption 1 that connected nodes in the graph are likely to share the same labels. Besides, there are many graph embedding studies [23, 42] built on top of Assumption 1 and 2 that the connected nodes are more likely to have the same class and similar node embeddings. Unfortunately, this assumption differs from reality in many nodes.

Reliable Edge Distillation is based on an intuition that when the two connected nodes have different labels, using a Graph Laplacian Regularization on the edge can highly degrade the learning. For the concerned edges, it is better not to use a Graph Laplacian Regularization. Therefore Reliable Edge Distillation imposes a strict limitation on when to use the edge: *only reliable edges should be used to express nodes adjacency information.* An edge between two nodes is reliable only if the nodes are themselves reliable and if their predicted labels belong to the same class.

More concretely, we improve the regularization item in Eq. 8 by exploiting the edge reliability. Based on the reliable edges, we minimize the following regularization loss:

$$\mathcal{L}_{\text{reg}} = \left\| f_t(\boldsymbol{x_i}) - f_t(\boldsymbol{x_j}) \right\|^2, \forall (v_i, v_j) \in \mathcal{E}_{\text{r}}. \quad (9)$$

4.2.3 ***Reliable Data Driven Optimization*** . After getting the reliable nodes and edges, we train the GCN with reliable data driven optimization. The optimization loss function for student models is in Eq. 10.

$$\mathcal{L} = \mathcal{L}_1 + \gamma \mathcal{L}_2 + \beta \mathcal{L}_{\text{reg}}, \quad (10)$$

where γ controls the proportion of reliable node knowledge h_t should transfer from the former teacher model H_{t-1}, and β controls the strength of the edge distillation. We use the standard Back Propagation (BP) algorithm [43] to optimize each GCN and its weights are updated with Eq. 10.

4.3 Graph Data Based Ensemble

After getting the student model h_t, we compute the information entropy of node v_i by(line 19 of Algorithm 3)

$$I_t(\boldsymbol{x_i}) = -\boldsymbol{h_t}(\boldsymbol{x_i}) \log \boldsymbol{h_t}(\boldsymbol{x_i}) \quad (11)$$

A low $I_t(\boldsymbol{x_i})$ means the h_t is confident on its prediction on node v_i. To evaluate the performance of each base model. A higher $Pr(\boldsymbol{x_i})$ means the node $\boldsymbol{x_i}$ is relatively more important since it has a high PageRank value and more nodes need its help to update their embeddings in the training process. So, we should give more attention to this node. Considering the importance of each node, we calculate the weights of each base model by (line 20 of Algorithm 3)

$$\alpha_t = \frac{1}{\sum_{i=1}^{N} I_t(\boldsymbol{x_i}) Pr(\boldsymbol{x_i})} \quad (12)$$

As the GCN can easily overfit while training on the labeled nodes, we consider the information entropy of the predictions given by h_t on both the labeled and unlabeled nodes, unlike traditional Boosting techniques which measure α_t by the accuracy on the labeled nodes. The higher α_t the more confident the model h_t is about its predictions, thus the more important its role in the final ensemble process.

After T iterations, we can get T student models. For each base model h_t, we can accordingly get its weight α_t, and then add it to the final ensemble teacher model H_t. More concretely, we average the softmax outputs of each base model h_t with the model weight α_t, and the ensemble model H_T is defined as (line 23 of Algorithm 3)

$$H_T = \sum_{i=1}^{T} \alpha_t h_t \quad (13)$$

5 EXPERIMENTS

5.1 Experimental settings

To validate the effectiveness of RDD, we performed an extensive evaluation in graph-based semi-supervised learning tasks on several real-world datasets. We first introduce the four datasets used in the experiments and then list the comparative baselines and their settings. Finally, we present the raw experimental results and discuss them.

Datasets. For comparison, we use the released partitioned datasets for the three citation networks as in [33] and the knowledge graph as in [36]. Table 2 displays an overview of these four datasets.

- **Citation networks** [33] produce node embeddings by truncating the Chebyshev polynomial to the first-order neighborhoods. Each node in the citation networks represents an article published in the corresponding journal. Edges between two nodes represent citations from one article to another, and labels represent the topic of the article. The feature vector of each node corresponds to a bag-of-word representation of the document. For the three citation datasets, 20 instances are sampled for each class as labeled data, 1000 instances as test data, and the rest are used as unlabeled

Table 2: Overview of the Four Datasets

Dataset	#Nodes	#Features	#Edges	#Classes
Cora	2708	1433	5429	7
Citeseer	3327	3703	4732	6
Pubmed	19717	500	44338	3
NELL	65755	61278	266144	210

data. We use an additional validation set of 500 labeled nodes for tuning hyperparameters as in [33].

- **Knowledge Graph NELL** NELL is a dataset extracted from the knowledge graph presented in [12]. For this dataset, each relation is described as a triplet (e_1, r, e_2) and will be assigned with separate relation nodes r_1 and r_2 as (e_1, r_1) and (e_2, r_2), where e_1 and e_2 are entities and r is the relation between them. As set in [33], we extend the features by assigning a unique one-hot representation for every node, which results in a 61278-dim sparse feature vector. For this semi-supervised task, we consider the label rates of 10% per class in the training set. An additional validation set of 500 labeled nodes is used for tuning hyperparameters and we do not use the labels for model training.

Baseline Methods. To evaluate the performances of the single model in RDD, we compare our method with four types of representative methods. First, we compare RDD with two representative graph-based SSL methods: **label propagation(LP)** [62] and **Planetoid** [54].

As RDD is an ensemble-based method, we first compare its final ensemble accuracy with other ensemble methods. Note that our method is not limited to the architecture of the base model. To make it fair, the base models of each ensemble method all use a two-layers GCN. We compare RDD with **Bagging** [9] and **BANs** [21].

Usually, the high computation cost renders ensemble models unsuitable for online prediction. The models are too large to fit into the main memory, especially for mobile phones. Still, a single model is much smaller than the ensemble model it contributing to and can be used as a classifier on its own. Especially the last single model in RDD has been trained under the supervision of the most powerful teacher model, hence it exhibits the best performances in terms of accuracy among all the base models. We compare the performances of the last single model in RDD with other start-of-the-art non-ensemble models: **GAT** [48], **GPNN** [36], **APPNP** [19], **LGCN** [22], **NGCN** [1], and **DGCN** [64].

Finally, some current methods aim at training a deep GCN to make full use of the unlabeled data. Since we have the same motivation, we also compare RDD with those: **ResGCN** [33], **Dense-GCN** [34] and **JK-Net** [51].

Settings. We use PyTorch to implement the following models: GCN, Bagging, BANs, JKNet, ResGCN, DenseGCN and our RDD models (both single and ensemble). For the other models (LP, Planetoid, LGCN, GPNN, NGCN, DGCN, APPNP and GAT), all the experimental results are drawn from their respective publications.

We train our models using Adam optimizer with a learning rate of 0.01 for each dataset. We set the l_2 regularization factor to 5e-4 for the citation networks and to 1e-5 for NELL. The dropout is applied to all feature vectors with rates of 0.8 to the citation networks and 0.2 to NELL. For the network architecture, the dimension of hidden features for the ResGCN and GCN is 16 for three citation networks and 100 for NELL. We increase the dimension of hidden features by 20 with each additional layer for JK-Net and DenseGCN. For example, the feature dimension of a 6 layers JK-Net is $\{90, 70, 50, 30, 10, F\}$, where F is the number of the classes of the given node classification task. Note that JK-Net has three aggregators and we choose the concatenation as the final aggregation layer since it performs best on the citation networks. We use the validation data to tune how many layers each method should use. Besides, for all ensemble methods, we train their base models with a two-layer GCN.

We tune three hyperparameters in RDD using the validation set. For each dataset, we set the parameter β to 10 and p to 40. Besides, to better transfer the knowledge of the teacher model, we proposed a cosine annealing method to adjust γ. Similar to SGDR [37], for the eth epoch in the total E training epochs, we adjust it as,

$$\gamma = \gamma_{initial} * \big(1 - \cos(e * \pi / E)\big), \tag{14}$$

We set the corresponding initial values of $\gamma_{initial}$ to 1,3,3 and 0.01 for the Cora, Citeseer, Pubmed and NELL. At the first of the training process, the prediction of the student model is inaccurate, so we should focus less on the $\mathcal{L}_2$ and $\mathcal{L}_{\text{reg}}$ loss. By using such a cosine annealing schedule, the student model converges faster.

For the training budget, we train every single model with 500 epochs and we terminate the training process if the validation accuracy does not improve for 20 consecutive steps. Besides, for each ensemble method, we train five base models and combine their outputs to get the final prediction. Note that we do not train base models in Bagging on the sampled data. That is because the labeled data in SSL is usually limited and sampling the dataset will introduce a high bias of base models due to the limited training data.

To eliminate random factors, we run each method 10 times and report the mean prediction accuracy on the test set. We use numbers to denote classification accuracy in percent.

5.2 Comparison to ensemble methods

We compare our RDD models –both the single model (labeled "RDD(Single)") and the ensemble model (labeled

Table 3: The accuracy comparison of RDD ensemble and single models with other ensemble methods on the citation and knowledge graph datasets. For each dataset, the best performance of the baselines are underlined. Not only our RDD ensemble model outperforms its competitors but the single models achieves highly competitive results.

Models	Cora	Citeseer	Pubmed	Nell
Single GCN	81.8	70.8	79.3	83.0
RDD(Single)	**84.8**	**73.6**	**80.7**	**85.2**
Bagging	84.2	72.6	80.1	85.1
BANs	84.5	72.1	79.8	85.4
RDD(Ensemble)	**86.1**	**74.2**	**81.5**	**86.3**

"RDD(Ensemble)")– to the other ensemble methods. The single regular GCN model is dubbed "Single GCN". After the ensemble of five base models, we compare the final ensemble accuracy, and the result is shown in Table 3. Table 3 shows how performs Reliable Data Distillation –both the single model and the ensemble model– compared to the two ensemble baselines: Bagging and BANs.

All the ensemble methods outperform the single GCN by at least 2.3%, 1.3%,0.5% and 2.1% on the four datasets respectively. It confirms that the use of an aggregation of multiple base models is less noisy and has its accuracy improved. While there is no clear winner between BANs and Bagging, the ensemble model of RDD provides the highest accuracy. For these four datasets, the ensemble RDD outperforms the other methods in all cases and provides gains from 0.9% to 1.6% in accuracy compared to BANs and Bagging. Surprisingly, the single RDD model also outperforms ensemble methods in three of the four datasets, showing the effectiveness of our approach.

Compared to BANs, RDD improves the diversity by preventing the student model to learn every prediction of the teacher model as ground truth, making the student more diverse than its teacher model than it would be otherwise. On the other hand, although Bagging has a high diversity by individually training each base model, its base models suffer from poor quality of prediction.

5.3 Comparison to single model

To demonstrate the performance of the single model, we compare the accuracy of the last base model of our method with other state-of-the-art methods. Table 4 shows the results obtained in terms of accuracy. Note that the best performance of each column is highlighted in boldface.

Our single RDD model outperforms the most competitive baseline (whose value is underlined), yielding significant improvement on each dataset: it achieves better performances

Table 4: Accuracy (in %) of the predictions of our RDD single model and its competitors on the citation networks. For each dataset, the best baseline values are underlined. Our model outperforms its competitors.

Models	Cora	Citeseer	Pubmed
LP	68.0	45.3	63.0
Planetoid	75.7	64.7	79.5
LGCN	83.3	73.0	79.5
GPNN	81.8	69.7	79.3
NGCN	83.0	72.2	79.5
DGCN	83.5	72.6	80
APPNP	83.3	71.8	80.1
GAT	83.0	72.5	79.0
GCN	81.8	70.8	79.3
RDD(Single)	**84.8**	**73.6**	**80.7**

Table 5: The accuracy comparison of our RDD models with deep GCN models. The best baseline is underlined for each dataset. Our models outperforms deep GCN models.

Models	Cora	Citeseer	Pubmed	Nell
GCN	81.8	70.8	79.3	83.0
JK-Net	81.8	70.7	78.8	84.1
ResGCN	82.2	70.8	78.3	82.1
DenseGCN	82.1	70.9	79.1	83.4
RDD(Single)	**84.8**	**73.6**	**80.7**	**85.2**

over the current state-of-the-art methods by a margin of 1.3%, 0.6%, and 0.6% on Cora (DGCN), Citeseer (LGCN), and Pubmed (APPNP), respectively. Note that the base model in RDD is GCN, and our single model exceeds the original GCN by 3.0%, 2.8%, and 1.4% when we train it on the Cora, Citeseer, and Pubmed datasets. While we chose GCN for its simplicity of implementation and its relatively low computational cost, our method is not limited to the base model we use, so the margin can be further improved if we use a more powerful base model like GAT [48].

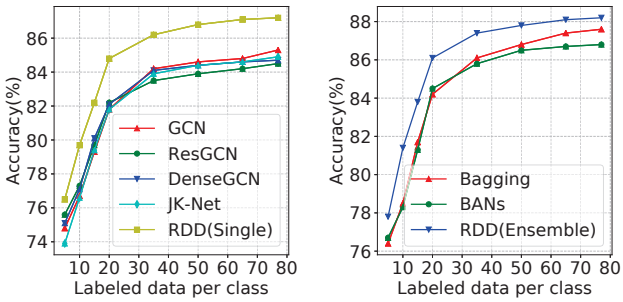
Compared with other baselines, every model in RDD cannot only be trained with a supervised loss but can also learn the reliable knowledge from a powerful teacher model to correct what it wrongly classifies. Through the comparison of the accuracy, we observe that RDD is capable of getting a high accuracy if we only use a single model, which is an important asset for mobile devices and online prediction.

5.4 Comparison to deep GCN

To make full use of the unlabeled data in a graph, a naive algorithm-based method is to train a deep GCN. RDD solves

Table 6: Impact of the use of ensemble technique on the Cora dataset. RDD single models are both accurate and diverse enough to benefit the most of ensemble technique.

Accuracy	Bagging	BANs	RDD(Ensemble)
Average	81.8	83.7	84.3
Ensemble	84.2	84.5	86.1
Gain	2.4	0.8	1.8



(a) Single model comparison (b) Ensemble model comparison

Figure 6: Performance of GCN using different number of labeled data on the Cora citation network

this problem from a data-driven perspective, and we compared it with ResGCN, DenseGCN, and JK-Net since all of them aim at training a deeper model. We gradually increase the layers of each deep GCNs and tune the number of layers for all compared methods on the validation dataset. Note that we report the best accuracy in all network architectures, and the raw results are displayed in Table 5.

ResGCN, DenseGCN, and JK-Net keep more information about the original features compared with GCN. However, their performances do exceed GCN by a small margin: they ignore the difference between each node and thus introduce the over-smoothing problem for the high degree nodes in the deep layers. For unlabeled nodes, How to train a deep GCN to make full use of their potential is still an open problem.

RDD outperforms all its competitors: its accuracy exceeds the next best competitor model by 2.6%, 2.7%, 1.4% and 1.1% in Cora, Citeseer, Pubmed and Nell respectively. So, RDD can make better use of the unlabeled nodes.

5.5 Analysis on Ensemble

To assess the effectiveness of our ensemble strategy, we first compute the average accuracy obtained by five base models of Bagging, BANs, and RDD. We compared that value to the corresponding ensemble accuracy. Table 6 shows the raw results and the gain for each method.

Among these three ensemble methods, Bagging gets the highest improved accuracy of 2.4%. Bagging trains each base model individually, thus it can get the highest diversity. However, the accuracy of every single model is low if we have only a few labeled nodes.

The single models of BANs have higher accuracy but, due to its KD learning which enforces the student model to mimic the knowledge of the teacher model, the diversity among the models is poor, leading to a limited increase in accuracy.

On the other hand, RDD takes the best of the two worlds: its use of reliability increases the diversity among the models while providing accurate single models. Not only RDD has the more accurate base models (+0.6% compared to BANs) but it also benefits from ensemble learning, leading to the best performances. RDD is thus the most suitable approach for GCN.

5.6 Analysis on graph sparsity

In this section, we evaluate the impact of the sparsity of the graph on the test accuracy. To change the sparsity, we change the number of labeled nodes per class. We compare RDD to its main competitors. For a fair comparison, we do not change the validation set and test set in Cora. We found each class has at least 77 labeled nodes in the training set, so we set the maximum number of labeled nodes per class to 77. We calculate the corresponding test accuracy when the labeled data per class is 5, 10, 15, 20, 35, 50, 65 and 77 respectively. The experiment results are shown in Figure 6.

As shown in Fig. 6(a), RDD(Single) always exceeds the compared baselines by a large margin when we increase the number of labeled data per class. This is because the single model in RDD can get extra supervision from the powerful ensemble model (teacher model). When we gradually increase the number of labeled data per class to 77, Fig. 6(b) shows the margin of RDD and Bagging decreases, which means the base models in Bagging also gain from the many labeled data per class. Both Bagging and RDD outperforms BANs when the number of labeled data per class is larger than 35. The student model in BANs mimics all the softmax outputs of teacher, leading to the low diversity of base models and an inaccurate ensemble model.

5.7 Exploration of hyperparameters

We investigate the impact of the hyperparameters on RDD. It has three hyperparameters: p controls the threshold of node reliability, γ controls the proportion of knowledge transfer and β controls the strength of the edge regularization. To evaluate their influence, we change their value and monitor the impact it causes. Note that we adjust γ using our defined cosine annealing method and the γ we use here means the

Table 7: Impact of the hyper-parameters on the accuracy (in %) on Cora dataset.

Parameters	p=40				p=80			
	$\gamma = 0$	0.5	1	1.5	$\gamma = 0$	0.5	1	1.5
$\beta = 0$	84.2	84.8	85.2	85.3	84.2	84.8	85.1	84.9
$\beta = 5$	84.5	84.7	85.4	85.2	84.4	84.9	85.0	85.1
$\beta = 10$	84.4	84.9	86.1	85.5	84.3	84.8	85.3	85.4
$\beta = 15$	84.6	84.7	85.8	85.3	84.5	84.5	85.2	85.1

initial value of γ. For these experiments, we focus on the Cora dataset. Table 7 shows the raw results.

For analyzing the influence of p, we set p to 40 and 80. We observe from Table 7 that RDD with the setting γ of 0.4 gets the higher accuracy in most cases. When setting a high value of p, more nodes and edges in a graph will be considered as reliable. Correspondingly, the student model in RDD can get more knowledge from the teacher model. However, the reliability of transferred knowledge may be reduced, which may introduce a student model with high bias. Besides, as the student mimics more knowledge from the teacher model, the diversity of the ensemble system may be reduced accordingly. As both the accuracy and diversity are important to an ensemble system, it is inappropriate to set a high value of p.

Setting γ to 0 means that each student model is trained without the L_2 loss function. In that case, the model only benefits from the reliable edge regularization and, as a result, the test accuracy of RDD suffers from an important drop. Similarly, if we set β to 0, the model will ignore the reliable edge regularization. In the extreme case where both γ and β are set to 0, every base model is trained individually: it is similar to Bagging.

Table 7 shows that the three parameters are important. The best combination on Cora to obtain the best accuracy of 86.1% is $p = 40$, $\gamma = 1$ and $\beta = 10$. While changing their values impact the accuracy, in the large majority of cases the results are still better than the ones of BANs and Baggin, showing the superiority of RDD over its competitors.

5.8 Impact of each contribution

To measure the impact of each contribution on the final accuracy, we test RDD while removing each feature at a time. We perform predictions with the obtained methods on the three citation networks datasets. We test RDD: *(i)* without the $\mathcal{L}_2$ loss function (called "No $\mathcal{L}_2$"), *(ii)* without the $\mathcal{L}_{\text{reg}}$ loss function (called "No $\mathcal{L}_{\text{reg}}$"), *(iii)* without our ensemble weighting scheme, i.e. use the same weighting scheme as Bagging (called "WEW"), *(iv)* without node reliability (called "WNR"), *(v)* without edge reliability (called "WER"), and *(vi)* without the knowledge reliability, i.e. without node and edge reliability (called "WKR"). Table 8 displays the results of the seven methods.

Table 8: Impact of the different key contributions of RDD on the accuracy (%). All of them have an important impact and cannot be removed without impacting the accuracy.

Method	Cora	Δ	Citeseer	Δ	Pubmed	Δ
No $\mathcal{L}_2$	84.4	-1.7	73.5	-0.7	80.2	-1.3
No $\mathcal{L}_{\text{reg}}$	85.2	-0.9	73.6	-0.6	80.9	-0.6
WNR	84.9	-1.2	73.3	-0.9	80.4	-1.1
WER	85.5	-0.6	73.4	-0.8	80.8	-0.7
WKR	84.8	-1.3	73.1	-1.1	79.8	-1.7
WEW	85.3	-0.8	73.7	-0.5	80.9	-0.6
RDD	**86.1**	–	**74.2**	–	**81.5**	–

Loss functions. The use of the L_2 loss function has a more important impact on the accuracy than the L_{reg} loss function. Knowledge transfer from the teacher knowledge plays is thus of the most importance.

Ensemble weighting scheme. Compared with the weighting scheme used in Bagging, our method improves the accuracy in all datasets. This is because we calculate the model weight according to node importance and prediction confidence.

Reliability cannot be overlooked. Reliability has a profound impact on the accuracy of the system. It worth noting that the node reliability contributes the most: for example on Cora when the node reliability is removed, the additional removal of the edge reliability decreases the accuracy by only 0.1%. This shows that, on the Cora dataset, most of the knowledge provided by the edge reliability is already contained in the node reliability. On PubMed, this value increases to 0.6%, showing that knowledge reliability cannot be reduced to the sole node reliability but that both notions are useful.

There is no clear winner between L_2 loss function and knowledge reliability, showing that both methods play an important part in RDD.

5.9 Efficiency analysis

Since RDD updates the reliable nodes and edges in each epoch, the training time will increase accordingly. It is necessary to analyze the extent to which the extra computation can affect efficiency. To achieve the accuracy of 84% on Cora, we report the training time each method needs on a GPU. The results are shown in Table 9.

Bagging has the fastest training time since both RDD and BANs rely on KD which requires a longer training time. RDD is slower due to its more elaborated training. RDD takes roughly twice the amount of time to train a single model.

Table 9: Training time using different ensemble methods on the cora dataset.

	Bagging	BANs	RDD(Ensemble)
Average time per model (s)	2.032	2.652	4.158
Number of base models	4	3	2
Total time (s)	8.128	7.956	8.316

However, as seen in Sec. 5.5, RDD needs fewer base models to reach an accuracy of 84% than its competitors: 4 for Bagging, 3 for BANS and only 2 for RDD. This leads to similar training time to achieve satisfactory performances.

6 RELATED WORKS

Graph data has been widely studied in recent years [28, 38, 40], and a large number of graph-based semi-supervised learning methods have been proposed since labeling an entire graph is extremely time-consuming. Among these methods, most of them make the cluster assumption that nearby nodes are likely to have the same labels [15]. Based on this idea, a series of works has been proposed in recent years. For example, we can learn a smooth low-dimensional embedding for each node with Markov random walks [45] and spectral kernels [58]. Besides, as a type of low-pass graph filtering [18], the label propagation [62] and its variants [5, 61] are also very popular.

The feature vectors of each node also contain much useful information, so many methods were proposed to jointly model graph structures and node features. A commonly used method is to regularize a supervised learner with the regularization. For example, Manifold regularization [3] exploits the geometry of the marginal distribution that generates the data and incorporates it as an additional regularization term. Besides, both deep semi-supervised embedding [49] and Planetoid [54] regularize a neural network with a Laplacian regularizer or an embedding-based regularizer.

Currently, the neural networks on the graph have attracted much attention due to its great performances [16, 53]. By introducing filters from the perspective of graph signal processing [44], a variant of graph convolution [10] is well designed for GCN. Accordingly, there are more and more improvements, expansions and approximations on spectral-based GCNs [50]. However, the spectral methods usually handle the whole graph simultaneously, and thus they are difficult to parallel or scale to large graphs.

To solve this problem, Spatial-based GCNs formulate graph convolutions as aggregating node features from neighbors. For example, Graphsage [24] samples the neighborhoods for each node, and then updates its feature by aggregating the adjacent information. Together with sampling strategies, the computation of Graphsage can be performed in a batch of nodes instead of the whole graph, thus it can be efficiently implemented in a large graph.

Based on these two types of GCN, many alternative GCNs have been proposed in the recent years, some representative methods include GAT [48], GPNN [36], APPNP [19], LGCN [22], NGCN [1] and DGCN [64]. However, due to the over-smoothing problem [35], these approaches cannot converge well using a deep architecture, thus limits the usage of the unlabeled nodes. Some recent algorithms are proposed to solve this problem, such as ResGCN [33], DenseGCN [34] and JK-Net [51].

Instead of finding a deep architecture, we consider this problem from a data perspective. Based on KD, we train each student model under the supervision of the teacher, and then the student can actively learn the transferred knowledge to correct what it learns incorrectly on both the labeled and unlabeled data. Compared with current GCNs, our method can take better advantage of the unlabeled nodes.

7 CONCLUSION

GCN is widely used in many applications, but it fails to use the full potential of the unlabeled nodes in a Semi-Supervised Learning task. In this paper, we proposed Reliable Data Distillation, a semi-supervised learning GCN learning method that makes better use of the unlabeled nodes. We introduced the notion of node and edge reliability in a graph. Using those notions, we designed a KD model, dubbed Reliable Data Distillation, which improves the learning by focusing on the learning of the student on reliable knowledge only. The obtained models are then combined into an ensemble learning method that outperforms the use of a single model. Our extensive evaluation on several real-world datasets demonstrated that Reliable Data Distillation –both the single and the ensemble models– outperforms its competitors by a significant margin on the node classification task.

ACKNOWLEDGMENTS

This work is supported by NSFC (No. 61832001, 61702015, 61702016, U1936104), the National Key Research and Development Program of China (No. 2018YFB1004403), Beijing Academy of Artificial Intelligence (BAAI), and PKU-Tencent joint research Lab. Lei Chen's work is partially supported by the Hong Kong RGC GRF Project 16214716, CRF Project C6030-18GF, AOE Project AoE/E-603/18, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Didi-HKUST joint research lab project, Microsoft Research Asia Collaborative Research Grant and Wechat and Webank Research Grant. Yingxia Shao is the corresponding author, and the first two authors contributed equally.

REFERENCES

[1] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. 2019. N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*. 310.

[2] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. *arXiv preprint arXiv:1905.00067* (2019).

[3] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7, Nov (2006), 2399–2434.

[4] Robert M Bell, Yehuda Koren, and Chris Volinsky. 2010. All together now: A perspective on the netflix prize. *Chance* 23, 1 (2010), 24–29.

[5] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. 11 label propagation and quadratic criterion. (2006).

[6] Iwo Białynicki-Birula and Jerzy Mycielski. 1975. Uncertainty relations for information entropy in wave mechanics. *Communications in Mathematical Physics* 44, 2 (1975), 129–132.

[7] Stephen Bonner, Ibad Kureshi, John Brennan, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. 2019. Exploring the Semantic Content of Unsupervised Graph Embeddings: An Empirical Study. *Data Science and Engineering* 4, 3 (2019), 269–289.

[8] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[9] Leo Breiman. 1996. Stacked regressions. *Machine learning* 24, 1 (1996), 49–64.

[10] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[11] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 535–541.

[12] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[13] Arjun Chandra, Huanhuan Chen, and Xin Yao. 2006. Trade-off between diversity and accuracy in ensemble generation. In *Multi-objective machine learning*. Springer, 429–464.

[14] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks* 20, 3 (2009), 542–542.

[15] Olivier Chapelle and Alexander Zien. 2005. Semi-supervised classification by low density separation.. In *AISTATS*, Vol. 2005. Citeseer, 57–64.

[16] Hongxu Chen, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Wen-Chih Peng, and Xue Li. 2019. Exploiting Centrality Information with Graph Convolutions for Network Representation Learning. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 590–601.

[17] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 785–794.

[18] Venkatesan N Ekambaram, Giulia Fanti, Babak Ayazifar, and Kannan Ramchandran. 2013. Wavelet-regularized graph semi-supervised learning. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 423–426.

[19] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[20] Fangcheng Fu, Jiawei Jiang, Yingxia Shao, and Bin Cui. 2019. An Experimental Evaluation of Large Scale GBDT Systems. *PVLDB* 12, 11 (2019), 1357–1370.

[21] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. *arXiv preprint arXiv:1805.04770* (2018).

[22] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1416–1424.

[23] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

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

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

[26] Thomas Hoch. 2015. An Ensemble Learning Approach for the Kaggle Taxi Travel Time Prediction Challenge.. In *DC@ PKDD/ECML*.

[27] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109* (2017).

[28] Pramod Jamkhedkar, Theodore Johnson, Yaron Kanza, Aman Shaikh, NK Shankaranarayanan, and Vladislav Shkapenyuk. 2018. A Graph Database for a Virtualized Network Infrastructure. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1393–1405.

[29] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-Supervised Learning With Graph Learning-Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11313–11320.

[30] Jiawei Jiang, Bin Cui, Ce Zhang, and Fangcheng Fu. 2018. DimBoost: Boosting Gradient Boosting Decision Tree to Higher Dimensions. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. 1363–1376.

[31] Jie Jiang, Jiawei Jiang, Bin Cui, and Ce Zhang. 2017. TencentBoost: A Gradient Boosting Tree System with Parameter Server. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. 281–284.

[32] Reza Khatami, Giorgos Mountrakis, and Stephen V Stehman. 2017. Mapping per-pixel predicted accuracy of classified remote sensing images. *Remote Sensing of Environment* 191 (2017), 156–167.

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

[34] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. 2019. Can GCNs Go as Deep as CNNs?. In *Proceedings of the IEEE International Conference on Computer Vision*. 29–38.

[35] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[36] Renjie Liao, Marc Brockschmidt, Daniel Tarlow, Alexander L Gaunt, Raquel Urtasun, and Richard Zemel. 2018. Graph partition neural networks for semi-supervised classification. *arXiv preprint arXiv:1803.06272* (2018).

[37] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).

[38] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. 2012. Distributed GraphLab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment* 5, 8 (2012), 716–727.

[39] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. 2018. Smooth Neighbors on Teacher Graphs for Semi-Supervised Learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 8896–8905.

[40] Anil Pacaci and M Tamer Özsu. 2019. Experimental Analysis of Streaming Algorithms for Graph Partitioning. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 1375–1392.

[41] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

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

[43] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation.* Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.

[44] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30, 3 (2013), 83–98.

[45] Martin Szummer and Tommi Jaakkola. 2002. Partially labeled classification with Markov random walks. In *Advances in neural information processing systems*. 945–952.

[46] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*. 1195–1204.

[47] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. 2018. Rgcnn: Regularized graph cnn for point cloud segmentation. In *Proceedings of the 26th ACM international conference on Multimedia*. 746–754.

[48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[49] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer, 639–655.

[50] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[51] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536* (2018).

[52] Yan Yan, Zhongwen Xu, Ivor W Tsang, Guodong Long, and Yi Yang. 2016. Robust semi-supervised learning through label aggregation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[53] Hongxia Yang. 2019. AliGraph: A Comprehensive Graph Neural Network Platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 3165–3166.

[54] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* (2016).

[55] Qi Ye, Changlei Zhu, Gang Li, Zhimin Liu, and Feng Wang. 2018. Using Node Identifiers and Community Prior for Graph-Based Classification. *Data Science and Engineering* 3, 1 (2018), 68–83.

[56] Minji Yoon, Jinhong Jung, and U Kang. 2018. Tpa: Fast, scalable, and accurate method for approximate random walk with restart on billion scale graphs. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1132–1143.

[57] Cha Zhang and Yunqian Ma. 2012. *Ensemble machine learning: methods and applications.* Springer.

[58] Tong Zhang and Rie Kubota Ando. 2006. Analysis of spectral kernel design based semi-supervised learning. In *Advances in neural information processing systems*. 1601–1608.

[59] Wentao Zhang, Jiawei Jiang, Yingxia Shao, and Bin Cui. 2020. Snapshot boosting: a fast ensemble framework for deep neural networks. *Sci. China Inf. Sci.* 63, 1 (2020), 112102.

[60] Kai Zhong, Guorui Feng, Liquan Shen, and Jun Luo. 2018. Deep learning for steganalysis based on filter diversity selection. *SCIENCE CHINA Information Sciences* 61, 12 (2018), 129105:1–129105:3.

[61] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.

[62] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.

[63] Xiaojin Jerry Zhu. 2005. *Semi-supervised learning literature survey.* Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[64] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 499–508.