Rethinking the Setting of Semi-supervised Learning on Graphs

Ziang Li[*], **Ming Ding**[*], **Weikai Li**, **Zihan Wang**, **Ziyu Zeng**, **Yukuo Cen**, **Jie Tang**
Department of Computer Science and Technology, Tsinghua University
{li-za19, dm18, liwk19, zhwang19, zengzy19, cyk20}@mails.tsinghua.edu.cn
jietang@tsinghua.edu.cn

Abstract

We argue that the present setting of semi-supervised learning on graphs may result in **unfair comparisons**, due to its potential risk of *over-tuning hyper-parameters* for models. In this paper, we highlight the significant influence of tuning hyper-parameters, which leverages the label information in the validation set to improve the performance. To explore the limit of over-tuning hyper-parameters, we propose ValidUtil, an approach to fully utilize the label information in the validation set through an extra group of hyper-parameters. With ValidUtil, even GCN can easily get high accuracy of 85.8% on Cora.

To avoid over-tuning, we merge the training set and the validation set and construct an i.i.d. graph benchmark (IGB) consisting of 4 datasets. Each dataset contains 100 i.i.d. graphs sampled from a large graph to reduce the evaluation variance. Our experiments suggest that IGB is a more stable benchmark than previous datasets for semi-supervised learning on graphs. Our code and data are released at https://github.com/THUDM/IGB/.

1 Introduction

Graph Neural Networks (GNNs) [Gori *et al.*, 2005; Kipf and Welling, 2016] have emerged as a heated field in machine learning in recent years. Among the widely known GNN models [Kipf and Welling, 2016; Hamilton *et al.*, 2017; Velickovic *et al.*, 2018; Feng *et al.*, 2020; Chen *et al.*, 2020], which one is the best? Most GNN papers demonstrate their performance on the task of semi-supervised learning on graphs following GCN, where a widely-used benchmark includes Cora, CiteSeer, and PubMed [Sen *et al.*, 2008].

This benchmark is competitive but unstable. For example, the accuracy of GCNII [Chen *et al.*, 2020] (SOTA method) on Cora is 85.5% with a dropout rate of 0.6, but it will drop to 79.0% if we slightly increase the dropout rate to 0.75. In contrast, the reported accuracy of GCN is about 81.5%.[1] Previous researchers attributed this kind of instability to the *small size* of the graphs and put forward larger benchmarks, e.g., OGB [Hu *et al.*, 2020] and HGB [Lv *et al.*, 2021]. However, to the best of our knowledge, few works challenge the *setting* of semi-supervised learning on graphs.

A similar predicament of the unstable performance also exists in few-shot natural language understanding, where Zheng *et al.* [2021] recently found that models were over-fitting the validation set via hyper-parameters. Since the labels in the validation set are even more than those in the training set, the searched value of hyper-parameters becomes vitally important. Inspired by this finding, we hypothesize that the same reason could also, to some extent, account for the instability of semi-supervised learning on graphs, since the size of the validation set is also usually much larger than that of the training set (e.g., 140 training samples vs. 500 validation samples in Cora) [Yang *et al.*, 2016]. Meanwhile, recent GNNs show a trend of owning more hyper-parameters. While GCN has very few hyper-parameters, GAT needs the accuracy on the validation set to determine its structures (e.g., the number of attention heads and the existence of a residual connection). PPNP [Klicpera *et al.*, 2019a] further requires a global diffusion radius and a teleport probability α. GDC [Klicpera *et al.*, 2019b] searches a diffusion radius and a threshold for sparsification as hyper-parameters on the validation set, which is improved in ADC [Zhao *et al.*, 2021] by replacing the grid-search with gradient-based optimization for layerwise and channel-wise diffusion radii. This evolving path of GNNs suggests that GNN models utilize the validation set to a greater and greater extent and that the more explicit utilization improves and stabilizes the training. This benchmark leads to an unfair comparison favoring models with larger sizes of hyper-parameters.

The same phenomenon still exists in OGB [Hu *et al.*, 2020], even though its percentage of validation set is much smaller than that of Cora. The participants [Wang *et al.*, 2021] find that directly merging validation set into the training set can significantly increase the performance, which is then only allowed on the collab dataset according to the updated OGB rules. Moreover, C&S [Huang *et al.*, 2020] incorporates the labels in the validation set during label propagation and obtains a great improvement. All of them indicate that simply reducing the validation set ratio or to increase the graphs' size is not a satisfying solution to the problem.

[*] indicates equal contribution.

[1] The experiments can be reproduced using the CogDL package [Cen *et al.*, 2021].

The findings urge us to rethink the setting of the semi-supervised learning on graphs and the meaning of validation set. On the one hand, the original motivation of introducing validation set is to optimize the hyper-parameters, which cannot be directly optimized by usual methods, e.g., stochastic gradient descent (SGD). On the other hand, we have only two kinds of samples in real-world applications, labeled and unlabeled. We have to split out a part of the labeled data as the validation set to search for the best hyper-parameters. This means it is a disadvantage instead of a merit to own a large set of hyper-parameters because they use a great quantity of labeled data as a validation set and result in a smaller training set providing real-world information. However, under the current setting, the extra and informative validation set encourages the model to equip itself with more hyper-parameters to fully utilize the labels in the validation set, which deviates from the real-world scenarios. In this work, we name the problem of "using hyper-parameters to fit validation labels" as **over-tuning**.

Present work. In this paper, we analyze the influence of the size of validation set and propose **ValidUtil**, a method to make any GNN fully utilize the label information in the validation set. **ValidUtil** explores the limit of over-tuning hyper-parameters. To avoid meaningless optimization towards the utilization of validation set and increase the stability of GNN benchmarks, we construct the i.i.d. graph benchmark (**IGB**) with the following two improvements:

- *Unify the training and validation set.* In IGB, the graphs have no pre-defined ratio of training and validation set but only labeled data and unlabeled data. Different models can freely split the labeled data into training and validation set based on the number of hyper-parameters. In this way, the over-tuning of hyper-parameters is discouraged.
- *Multiple i.i.d. graphs and diverse domains.* We have 4 datasets consisting of a co-authorship network, a social network, a knowledge graph, and a photo sharing network. In each dataset, we sample 100 subgraphs using a modified Random-Walk method. The sampled graphs are approximately i.i.d, while each gives a reliable evaluation of GNN performance. Therefore, we can obtain more stable metrics by averaging performances over them.

2 The Risk of Over-tuning of Semi-supervised Learning on Graphs

2.1 Semi-Supervised Learning on Graphs

Definition. Given an undirected graph $G = (V, E)$, where the node set V contains n nodes $\{v_1, ..., v_n\}$ and E is the edge set. Each node v_i is associated with a feature vector $\mathbf{x}_i$ and a class label y_i. We denote the set of node labels as $\mathbf{Y}$. In the task of semi-supervised node classification on graphs (transductive), only a small part of node labels $\mathbf{Y}^L \subset \mathbf{Y}$ are given, and the rest label set $\mathbf{Y}^U = \mathbf{Y} - \mathbf{Y}^L$ needs to be predicted. Usually $|\mathbf{Y}^L| \ll |\mathbf{Y}^U|$.

Here, we briefly introduce three widely used citation networks (i.e., Cora, CiteSeer, PubMed) for the analysis in this section. In these datasets, node features are bag-of-words representations of documents. Each dataset is a connected graph constructed based on the citation links between documents. Each dataset uses 20 training samples per class as labeled data in the semi-supervised setting. Table 1 shows the statistics of the three datasets.

Dataset	Nodes	Edges	Split	Classes	Features
Cora	2,708	5,429	140 / 500 / 1,000	7	1,433
CiteSeer	3,327	4,732	120 / 500 / 1,000	6	3,703
PubMed	19,717	44,338	60 / 500 / 1,000	3	500

Table 1: Statistics of Cora / CiteSeer / PubMed datasets.

2.2 An Analysis of Over-tuning in Current GNNs

In this section, we will investigate the over-tuning phenomenon in current GNNs. As discussed above, the hyper-parameters act as a tool to utilize the labels in the validation set. Therefore, the models' performance should improve as we increase the size of the validation set. We select five representative GNNs, GCN, GAT, APPNP, GDC-GCN, and ADC, and exhibit their accuracy on Cora with different sizes of validation set. The search scope of the hyper-parameters includes learning rate, hidden size, early stopping iteration, number of layers, the dropout rate, the diffusion radius of APPNP and GDC, the sparsification threshold of GDC, etc. We use grid search to find the best hyper-parameters for each model. Details about the search scopes are shown in the released codes.

We run the experiments on the public split [Yang *et al.*, 2016] of Cora, ranging the size of validation set from 10 to 500 by hiding a part of the labels. For each validation size, we report the accuarcy on the test set after training the model with the best searched hyper-parameters. The results are demonstrated in Figure 1.

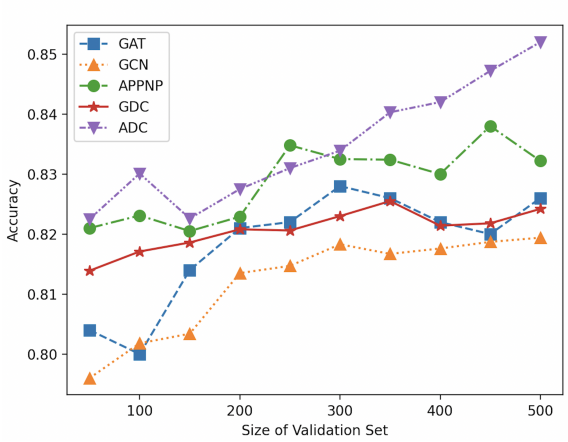

Figure 1: The accuracy of models with different size of the validation set on Cora. The test accuracy is the average of 20 runs with different random seeds.

Figure 1 shows that the GNN models have a clear trend that the performance is usually better with a larger validation

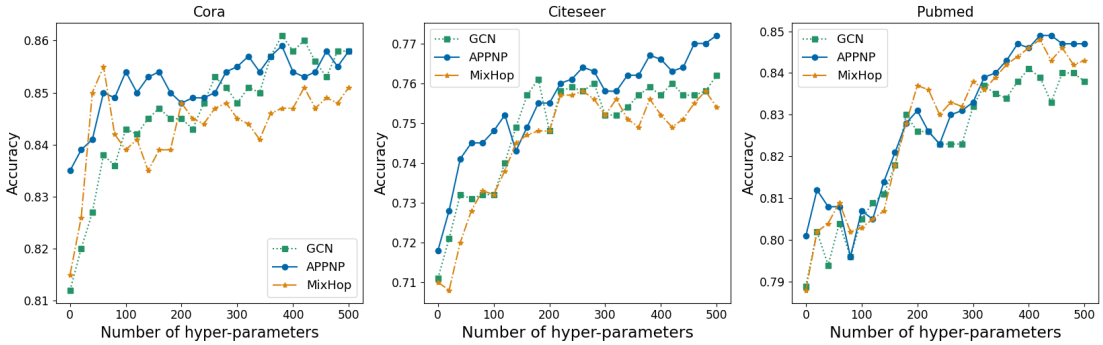

Figure 2: The test accuracy of ValidUtil plus GCN, GAT, MixHop and PPNP on Cora, Citeseer, and PubMed. The horizontal axis means the number of new hyper-parameters from ValidUtil, where 0 is equivalent to the original GNN without ValidUtil.

set. Since the validation set can only affect the model via the hyper-parameters, we can conclude that the model benefits from the validation labels with the help of hyper-parameters. The accuracy improvement is up to $1\% \sim 3\%$ if we increase the size of validation set from 100 to 500, which is significant enough to suggest that the over-tuning already exists.

2.3 ValidUtil: Exploring the Limits of Over-tuning

Although the analysis above shows that over-tuning influences GNN models' performance, we wonder to what extent the influence could achieve. If the importance of validation labels is much smaller than that of model structures, the previous benchmarks will still be a proper choice for GNN benchmarks. If not, we should rethink and re-design the evaluation pipeline and datasets for semi-supervised learning on graphs.

The most intuitive method to fully utilize the validation labels is to merge the validation set into the training set. However, this operation is universally acknowledged as invalid and is forbidden since it is a kind of data leakage. In this section, we put forward ValidUtil, a technique to mimic this operation via searching hyper-parameters. ValidUtil is *not* really a method to improve GNNs, but more like a "reduction to absurdity". The full pipeline of ValidUtil is defined as follows and in Algorithm 1:

1. **Add hyper-parameters.** For any given model, we add t extra hyper-parameters $\mathbf{Y}^T = \{\hat{y}_1, ..., \hat{y}_t\}$, which refers to "pseudo-labels of the t nodes in the validation set". These hyper-parameters affect the model training in a way that they act as a known label for the corresponding validation node; that is to say, the model will be trained on an augmented training set $\mathbf{Y}^L \cup \mathbf{Y}^T$. And since the labels of the validation set given to the model are not ground-truth labels, this is not a data leakage process.
2. **Alternately optimize hyper-parameters.** The most common search method for hyper-parameters is grid search, but it is time-consuming when there are many hyper-parameters. Thus, we alternately search the best value of each $\hat{y}_i$ individually. In the beginning, each $\hat{y}_i$ is initialized as the predicted label y'_i of the GNN without ValidUtil. We will search and fix the best $\hat{y}_i$ one by one, following the method described in the next paragraph. We set the dropout rate as 0 when searching for the best pseudo-labels.
3. **Search the best pseudo-label for each validation node.** To search for the best value of $\hat{y}_i$, we enumerate all possible values of $\hat{y}_i$ while keeping the other hyper-parameters unchanged. For each possible value, we train a GNN model using labels $\mathbf{Y}^L \cup \mathbf{Y}^T$ and select the best value according to the accuracy of the validation set. This is the standard process of searching a hyper-parameter.
4. **Train the final model based on the best pseudo-labels.** After the pseudo-labels are determined, we can train on $\mathbf{Y}^L \cup \mathbf{Y}^T$, and use ordinary grid search to determine other hyper-parameters, including the dropout rate, and

Algorithm 1 ValidUtil

Input Graph G with nodes in training/validation/test set. We denote their labels as $Y^L/Y_{valid}/Y_{test}$. There are k classes.
Output Accuracy of the GNN model on test set.

Train a GNN model M with Y^L.
Predict the labels of validation set with M as $Y^P = \{y'_1, ..., y'_t\}$.
Add t extra hyper-parameters $Y^T = \{\hat{y}_1, ..., \hat{y}_t\}$.
Initialize $Y^T = Y^P$.
for i from 1 to t **do**
    for l from 1 to k **do**
        $\hat{y}_i = l$.
        Train a GNN M with $Y^L \cup Y^T$.
        Acc_l = the accuracy of M on validation set.
    end for
    $l_{max} = \arg\max_{l \in \{1,...,k\}} Acc_l$.
    if $Acc_{l_{max}} > Acc_{l_{y'_i}}$ **then**
        $\hat{y}_i = l_{max}$
    else
        $\hat{y}_i = y'_i$
    end if
end for
Train a new GNN M with $Y^L \cup Y^T$.
return the accuracy of M on test set.

report the results on the test set.

Analysis of Effectiveness. The key reason for the effectiveness of ValidUtil is that in most cases, we can obtain the true label y_i for validation node v_i in step 3, which makes the final training equivalent to the training on the union of training and validation set. If the model is over-parameterized, it is powerful enough to overfit the predicted label of v_i as the true label y_i after sufficient training [2]. Then in most cases, the highest accuracy is reached if and only if $\hat{y}_i = y_i$. We find that most GNNs models are powerful enough to overfit the pseudo-labels on Cora, Citeseer, and PubMed in practice.

	Cora	Citeseer	PubMed
GCNII (sota Cora)	**85.5**	73.4	80.3
GRAND (sota Citeseer)	85.4	**75.4**	82.7
SAIL (sota PubMed)	84.6	74.2	**83.8**
GCN + ValidUtil	**85.8**	76.0	83.8
MixHop + ValidUtil	84.9	75.5	84.2
PPNP + ValidUtil	**85.8**	**77.3**	**84.7**

Table 2: Comparison between ValidUtil and the sota methods on Cora, Citeseer, and PubMed.

We demonstrate the performance of ValidUtil plus three GNN models, GCN, PPNP, and MixHop, in Figure 2. We find that even only $20 \sim 60$ hyper-parameters from ValidUtil can bring about a leap in performance for some models. When we add hyper-parameters for all the 500 nodes in the validation set, PPNP can achieve an accuracy much better than that of the sota methods in Table 2.

Remark. Although ValidUtil works purely by utilizing the validation labels, it is totally valid under the current setting. If we treat the GNN+ValidUtil as a black-box model, the training process is quite normal. ValidUtil actually utilizes the labels with low efficiency, because each hyper-parameter can only learn the information of one node – but this is enough to verify our hypothesis. **The current setting cannot prevent the validation labels from "leaking" during hyper-parameters tuning.** We believe that there exist some more efficient ways to define *influential* hyper-parameters. These hyper-parameters could be entangled with the features or model structures, and they can acquire information from multiple validation labels. According to Figure 1, such influential hyper-parameters might already exist in some models and cannot be easily detected. Therefore, it is urgent to construct a new benchmark for semi-supervised learning on graphs to avoid over-tuning and fairly and robustly compare GNN models.

[2]Some weak GNNs, e.g. vanilla GCN and GAT on CiteSeer, cannot well distinguish nodes in a strongly connected graph, and therefore cannot overfit the given labels. We add an additional self-loop for each node for GCN to solve the problem. This trick is already implemented in PyG [Fey and Lenssen, 2019] by passing `improved=True` for GCNs.

3 IGB: An Independent and Identically Distributed Graph Benchmark

3.1 Overview

Our new benchmark has two aims: avoiding over-tuning and being more robust.

To avoid over-tuning, we propose a new setting where there are only two sets of nodes, labeled and unlabeled. The model can use the labeled set in any way to train the best model and evaluate its performance on the unlabeled (test) set. If we need to search hyper-parameters, we can split out a part of the labeled nodes as the validation set. The over-tuning problem is eliminated because the validation labels are already exposed. This setting is closer to real-world scenarios and enables fair comparison between models with different sizes of hyper-parameters. To easily migrate the GNNs to this new setting, we will introduce a simple and powerful method to create validation set in section 3.2.

To construct a more robust benchmark, we expect models' performances to be stable for different random seeds. One of the most common methods in machine learning to reduce the variance of evaluation results is to test repeatedly and report the average performance. To achieve that, we expect to test a model's performance on multiple i.i.d graphs. However, how can we get multiple i.i.d. Cora-like graphs to evaluate the results?

If we think about the construction of the citation networks such as Cora and Citeseer, we will find that the papers are crawled down by spiders from the Internet, which means these networks can be seen as sampled from the large real-world citation network. A similar assumption is already used in previous works [Yang *et al.*, 2020] that the real-world graph data are sampled from a large underlying graph. To acquire i.i.d. graphs, we could sample "again" from an established graph. With appropriate sampling strategies, we can construct a group of i.i.d. graphs. The details about sampling are introduced in section 3.4.

3.2 The Pipeline of Evaluation

To solve the over-tuning problem, we have to update the pipeline of the task of semi-supervised learning on graphs. Following the real-world scenarios, we only split the nodes in a graph into two sets, labeled and unlabeled (with a ratio of 1:4 by default in IGB). The model can use the labeled set in any way to train the best model, and evaluate its performance on the unlabeled (test) set. A recommended method is as follows:

1. Divide the labeled set into training and validation sets.[3]
2. Find the best hyper-parameters using grid search on the training and validation sets from the first step.
3. Train the model with the best hyper-parameters on the full labeled nodes.
4. Test the performance of the model from the third step on the unlabeled (test) sets.

[3]The best ratio may differ from model to model. In practice, we find that 1:1 is an appropriate ratio for most models.

5. Repeat the above steps on each graph in a dataset and report the average accuracy.

The first two steps aims to find the best hyper-parameters for the GNN model. We believe that this approach is suitable for many GNN models to get satisfying hyper-parameters. If there are other reasonable methods to decide the best hyper-parameters with the labeled set, they will also be encouraged to replace the first two steps in this pipeline. In this way, we can avoid over-tuning by directly exposing all the label information in the validation set later in the third step.

3.3 Datasets

IGB consists of four datasets: AMiner [Tang *et al.*, 2008], Facebook [Rozemberczki *et al.*, 2019], NELL [Yang *et al.*, 2016], and Flickr [Zeng *et al.*, 2019]. Each dataset contains 100 undirected connected graphs, sampled from the original large graph according to the random walk method in section 3.4. We also report the average node *overlap rate*, the ratio of common nodes to the total size of nodes for a pair of sampled graphs. The coverage rate is defined as the ratio of the union of the 100 sampled graph to the original large graph. Lower overlap rate and higher coverage rate are preferred. The statistics of the datasets are reported in Table 3.

	AMiner	Facebook	NELL	Flickr
Average Nodes	4,485 $\pm$ 26	3,475 $\pm$ 71	3,540$\pm$ 68	4,452 $\pm$ 31
Edges	5,000	5,000	5,000	5,000
Features	3,883	128	10,000	500
Classes	8	4	164	7
Original Size	236,017	22,470	63,910	89,250
Overlap Rate	0.083	0.339	0.146	0.119
Coverage Rate	0.550	0.958	0.956	0.969

Table 3: Statistics of the datasets in IGB.

AMiner. The AMiner dataset is a co-authorship graph extracted from the AMiner system [Tang *et al.*, 2008]. Nodes represent authors, while edges mean co-authorship in at least one paper. Node features indicate the venues in which the author has publications. Specifically, each feature has 3,883 dimensions, and each dimension is 0 or 1, representing whether or not an author has had publications in the corresponding venue. Node labels represent the authors' main research fields.

Facebook. The Facebook dataset is the Facebook Page-Page dataset from the paper [Rozemberczki *et al.*, 2019]. It is a graph of official Facebook pages. Nodes are official Facebook pages, while edges are mutual likes between the pages. Node features are extracted from the page descriptions. Node labels are one of the following 4 categories defined by Facebook: politician, governmental organization, television show, and company.

NELL. The NELL dataset is a knowledge graph dataset generated from the NELL knowledge graph [Carlson *et al.*, 2010]. Nodes represent entities, and edges represent relationships between two entities. Each node initially has a 61,278-dimension feature, a binary bag-of-words representation of entity descriptions. We only reserve the features of the most frequent 10,000 words for efficiency.

Flickr. The Flickr dataset is a graph of photos uploaded to the Flickr website. Each node represents a photo, and an edge means two photos share some properties in common, such as from the same location or the same gallery. 500-dimension node features are bag-of-words representations of photos. The labels are one of 7 classes developed by Zeng *et al.* from 81 original tags.

3.4 Sampling Algorithm

The simplest way to make the subgraph's node label distribution similar to that of the original graph is vertex sampling. However, it does not meet our expectations because it generates unconnected subgraphs. To obtain nearly i.i.d. subgraphs for our benchmark, we must carefully design the sampling strategy and principles. Specifically, we expect the sampling strategy to have the following properties:

1. The sampled subgraph is a connected graph.
2. The distribution of the subgraph's node labels is close to that of the original graph.
3. The distribution of the subgraph's edge categories (edge category is defined by the combination of its two endpoints' labels) is close to that of the original graph.

The first property can be well satisfied by the Random-Walk (RW) algorithm. When performing RW on an undirected graph $G = (V, E)$, we start sampling from node $u = n_0$, and the following nodes can be selected by the transition possibility:

$$P_{u,v} = \begin{cases} \frac{1}{d_u}, & if\ (u,v) \in E, \\ 0, & otherwise, \end{cases}$$

where $P_{u,v}$ is the transition possibility from node u to v, and d_u is the degree of node u.

We retreat to *reject sampling*-like methods to guarantee the second and third properties. Here we introduce the Kullback–Leibler divergence (KL divergence) as a metric to measure the difference between two different distributions. Aiming to get 100 subgraphs with relatively low KL divergence of node labels' distribution ("Node KL") and edge categories' distribution ("Edge KL"), we set a pre-defined threshold to decide whether to accept a sampled subgraph or not. The comparison of the results before and after adding the threshold is shown in Table 4.

	AMiner	Facebook	NELL	Flickr
Node KL	0.0186 $\pm$ 0.0107	0.1306 $\pm$ 0.0427	0.4393 $\pm$ 0.1008	0.0060 $\pm$ 0.0025
Edge KL	0.0189 $\pm$ 0.0149	0.0284 $\pm$ 0.0121	0.2796 $\pm$ 0.0678	0.0046 $\pm$ 0.0015
+Thresholds				
Node KL	0.0123 $\pm$ 0.0024	0.0326 $\pm$ 0.0064	0.3184 $\pm$ 0.0243	0.0041 $\pm$ 0.0006
Edge KL	0.0062 $\pm$ 0.0008	0.0243 $\pm$ 0.0120	0.2068 $\pm$ 0.0179	0.0021 $\pm$ 0.0003

Table 4: KL divergence of the sampling results.

3.5 Benchmarking Results

We evaluate 7 representative GNNs on IGB: Grand [Feng *et al.*, 2020], GCNII [Chen *et al.*, 2020], APPNP [Klicpera *et al.*, 2019a], GAT [Velickovic *et al.*, 2018], GCN [Kipf and Welling, 2016], GraphSAGE [Hamilton *et al.*, 2017] and

	AMiner	Facebook	NELL	Flickr	Avg
Grand	82.5±0.8	88.9±1.0	84.4±1.1	44.3±0.8	75.0
GCN	76.5±1.1	87.9±1.0	93.9±0.7	41.9±1.3	75.1
GAT	78.8±1.0	88.3±1.2	91.1±1.2	43.1±1.3	75.3
GraphSAGE	81.6±0.8	87.2±1.1	**94.9±0.6**	43.4±0.9	76.8
APPNP	87.0±1.0	88.0±1.3	93.0±0.8	44.6±0.9	78.2
MixHop	86.1±1.1	89.1±0.9	94.7±0.7	43.5±1.2	78.4
GCNII	**88.4±0.6**	**89.5±0.9**	91.5±1.0	**44.7±0.8**	**78.5**

Table 5: Evaluation Results of GNNs on IGB.

MixHop [Abu-El-Haija *et al.*, 2019]. The results are shown in Table 5.

To evaluate the GNN model, we first define a search scope for each hyper-parameter. The scope is carefully chosen in order to include the best values on all the datasets. For each model, the best hyper-parameters in its original paper and the best hyper-parameters reported by CogDL [Cen *et al.*, 2021] are usually included in the search scopes. After that, we use our IGB benchmark to evaluate each model under the setting introduced in the section 3.2.

3.6 The Stability of IGB

We verify the stability of IGB in two ways. Firstly, we verify its stability when evaluating models on different graphs, as each IGB dataset contains 100 nearly i.i.d. graphs. Specifically, we compare the variances of the accuracies on 100 AMiner's subgraphs (IGB style) and on 100 Cora's random data splits (Cora style). The result shown in Figure 3 strongly suggests that the evaluation on IGB is more stable than Cora style, even though each AMiner graph uses random data split.

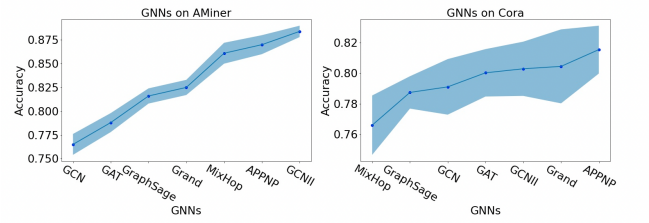

Figure 3: Accuracy of GNNs on AMiner and Cora. The blue area is the fluctuation range of the test accuracy. The results are based on 100 runs on AMiner's subgraphs or Cora's random splits.

Secondly, we focus on IGB's stability when evaluating models with different random seeds. In a stable benchmark, the ranks of different models should not easily change when changing the random seed. To verify this, we use the "inversion number" [4] of the ranking as a metric. Specifically, we evaluate the seven models using ten different random seeds, providing ten ranking sequences $S_i (i \in [1, 10])$. On each ranking sequence, we sort models according to their accuracy. The ranking sequence of the first seed is used as a *reference* sequence $S_1 = \{m_1, m_2 ... m_7\}$, where m_i is a GNN model. We call $m_j > m_k$ if and only if m_j ranks higher than m_j on S_1. For another sequence $S_i, i \neq 1$, if $m_j > m_k$, but m_j ranks lower than m_k on S_i, we call (j, k) an inversion pair in S_i. "Inversion number" is the number of inversion pairs in all sequences $S_i, i \neq 1$. Therefore, a high "inversion number" indicates a high instability of evaluation with different seeds. The result is reported in Table 6. IGB has a significantly smaller inversion number than Cora, CiteSeer, and PubMed, demonstrating its strong stability.

[4]The definition of "inversion number" can be seen in Wikipedia.

Cora	CiteSeer	PubMed	AMiner	Facebook	NELL	Flickr
67	45	107	0	9	0	5

Table 6: The inversion numbers of the ranking sequences using 10 different random seeds. Smaller inversion number indicates better stability.

4 Discussion

Is limiting the number of hyper-parameters a good way to solve the over-tuning problems? In section 2, we illustrate the power of over-tuning, where the improvements basically correlate with the number of hyper-parameters. However, if we set a hard limit for the number of hyper-parameters, complicated optimizers with many hyper-parameters, e.g. Adam [Kingma and Ba, 2014], will not be encouraged due to this limit. The models will also be encouraged to investigate more influential hyper-parameters to utilize the labels in the validation set under the limited budget of hyper-parameters. Therefore, the most fundamental solution is to change the evaluation setting as IGB.

What is the best GNN? In the results of IGB, GCNII performs the best. However, the performance differs in different datasets. For example, the sota method on Citeseer, GRAND, performs badly on NELL and thus gets a low average score, because NELL is a knowledge graph, whose distribution is quite different from that of citation networks. Will a GNN be good at all kinds of graphs, or do we need to design different GNNs for different categories of graphs?

5 Conclusion

In this paper, we revisit the setting of semi-supervised learning on graphs, identify the over-tuning problem, and prove its significance via the experiments of ValidUtil. To solve it, we propose a new benchmark, IGB, with a more reasonable evaluation pipeline. To further increase evaluation stability, we propose a sampling algorithm based on RW. GNNs are evaluated on the new benchmark, and the results demonstrate a stable performance rank. We expect that IGB can benefit the graph learning community by stabilizing the evolving path of GNNs in the future.

References

[Abu-El-Haija *et al.*, 2019] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, 2019.

[Carlson *et al.*, 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom Michael Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[Cen *et al.*, 2021] Yukuo Cen, Zhenyu Hou, Yan Wang, Qibin Chen, Yizhen Luo, Xingcheng Yao, Aohan Zeng, Shiguang Guo, Peng Zhang, Guohao Dai, et al. Cogdl: An extensive toolkit for deep learning on graphs. *arXiv preprint arXiv:2103.00959*, 2021.

[Chen *et al.*, 2020] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

[Feng *et al.*, 2020] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, and Jie Tang. Graph random neural network for semi-supervised learning on graphs. In *NeurIPS*, 2020.

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

[Gori *et al.*, 2005] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IJCNN*, volume 2, pages 729–734. IEEE, 2005.

[Hamilton *et al.*, 2017] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *ArXiv*, abs/2005.00687, 2020.

[Huang *et al.*, 2020] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[Klicpera *et al.*, 2019a] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

[Klicpera *et al.*, 2019b] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *NeurIPS*, 2019.

[Lv *et al.*, 2021] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *KDD*, 2021.

[Rozemberczki *et al.*, 2019] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *CoRR*, abs/1909.13021, 2019.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29:93–106, 2008.

[Tang *et al.*, 2008] Jie Tang, Jing Zhang, Limin Yao, Juan-Zi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *KDD*, 2008.

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

[Wang *et al.*, 2021] Zhitao Wang, Yong Zhou, Litao Hong, Yuanhang Zou, and Hanjing Su. Pairwise learning for neural link prediction. *arXiv preprint arXiv:2112.02936*, 2021.

[Yang *et al.*, 2016] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

[Yang *et al.*, 2020] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *KDD 2020*, pages 1666–1676, 2020.

[Zeng *et al.*, 2019] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. *CoRR*, abs/1907.04931, 2019.

[Zhao *et al.*, 2021] Jialin Zhao, Yuxiao Dong, Ming Ding, Evgeny Kharlamov, and Jie Tang. Adaptive diffusion in graph neural networks. In *NeurIPS*, 2021.

[Zheng *et al.*, 2021] Yanan Zheng, Jing Zhou, Yujie Qian, Ming Ding, Jian Li, Ruslan Salakhutdinov, Jie Tang, Sebastian Ruder, and Zhilin Yang. Fewnlu: Benchmarking state-of-the-art methods for few-shot natural language understanding. *arXiv preprint arXiv:2109.12742*, 2021.