Graph Neural Networks Without Training: Harnessing the Power of Labels as Input Features

Abstract

This study introduces a novel concept of training-free graph neural networks (TFGNNs) for transductive node classification, which can function immediately without any training and can optionally be enhanced through subsequent training. Initially, we put forward the idea of using labels as features (LaF), a valid yet relatively unexplored method in graph neural networks. Our analysis demonstrates that incorporating labels as features significantly improves the representational capacity of GNNs. The design of TFGNNs is based on these findings. Empirical evaluations show that TFGNNs surpass current GNNs in scenarios where training is not performed, and when training is optionally applied, they achieve convergence much faster than conventional GNNs.

1 Introduction

Graph Neural Networks (GNNs) have gained prominence as effective models for handling graph-structured data. They have demonstrated impressive performance across a range of tasks, including chemical structure analysis, question answering systems, and recommender systems.

A common application for GNNs is transductive node classification. In this task, the objective is to infer the labels of specific nodes within a graph, given the labels of other nodes. This approach finds utility in various real-world scenarios, such as classifying documents, analyzing e-commerce data, and studying social networks. Several GNN architectures, including Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), have successfully addressed transductive node classification, yielding excellent results.

A significant hurdle in the practical application of GNNs is their computational demand. Real-world graphs, such as those representing social networks or the structure of the web, can be enormous, containing billions of nodes. Processing these massive graphs can be computationally prohibitive. While various methods have been developed to enhance the efficiency of GNNs, such as node and edge sampling techniques, these methods still necessitate numerous training iterations. Other approaches, like PinSAGE, utilize parallel training and importance pooling to accelerate the training process, but they demand substantial computational resources. Consequently, the immediate deployment of GNNs with limited resources remains a challenge.

In this work, we introduce the concept of training-free graph neural networks (TFGNNs). To realize TFGNNs, we first propose the innovative idea of using labels as features (LaF). In the context of transductive node classification, utilizing node labels as features is a permissible approach. GNNs employing LaF can leverage label information, like the distribution of classes among neighboring nodes, to generate node embeddings. These embeddings are richer in information compared to those derived solely from node features. We establish that incorporating labels as features demonstrably augments the expressive capability of GNNs.

TFGNNs possess the unique ability to operate without any training, enabling immediate deployment upon initialization. This eliminates the need for extensive hyperparameter tuning when used in training-free mode. Furthermore, TFGNNs can be refined through optional training. Users have the flexibility to employ TFGNNs without training or to train them for a limited number of iterations when computational resources are constrained. This adaptability is particularly valuable in online learning scenarios, where data arrives sequentially, and the model needs to be updated promptly. TFGNNs can also undergo full training when resources are plentiful or when higher accuracy is paramount. In essence, TFGNNs offer the advantages of both nonparametric models and traditional GNNs.

Our experiments confirm that TFGNNs surpass existing GNNs when used without training and achieve convergence significantly faster than traditional GNNs when training is applied.

The primary contributions of this research are outlined below:

* We propose the utilization of labels as features (LaF) in transductive learning settings. * We provide formal proof that LaF enhances the representational power of GNNs. * We introduce a novel architecture for training-free graph neural networks (TFGNNs). * We empirically demonstrate that TFGNNs outperform existing GNNs in the absence of training.

2 Background

2.1 Notations

For any positive integer n, $[n]$ represents the set $\{1, 2, ..., n\}$. A graph is represented by a tuple comprising (i) a set of nodes V, (ii) a set of edges E, and (iii) node features $X = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^{n \times d}$. We assume nodes are numbered from 1 to n. Y denotes the set of possible labels. $y_v \in \mathbb{R}^{|Y|}$ is the one-hot encoded label for node v. $N(v)$ represents the set of neighboring nodes of node v. We use numpy-like indexing notation. For instance, $X_{:,1}$ denotes the first column of X, $X_{:,-1}$ denotes the last column, $X_{:,-5:}$ denotes the last five columns, and $X_{:,:-5}$ denotes all columns except the last five.

2.2 Transductive Node Classification

Problem (Transductive Node Classification). **Input:** A graph $G = (V, E, X)$, a set of labeled nodes $V_{train} \subset V$, and the corresponding labels $Y_{train} \in Y^{V_{train}}$ for these nodes. **Output:** Predicted labels $Y_{test} \in Y^{V_{test}}$ for the remaining nodes $V_{test} = V \setminus V_{train}$.

The node classification problem has two distinct settings: transductive and inductive. In the transductive setting, a single graph is provided along with the labels for a subset of its nodes, and the task is to predict the labels for the unlabeled nodes within the same graph. This contrasts with the inductive setting, where separate graphs are used for training and testing. For example, in the context of spam detection, if we label spam accounts on a social network like Facebook and then use a trained model to identify spam accounts on the same network, this is a transductive scenario. Conversely, if we use the model trained on Facebook data to identify spam accounts on a different platform like Twitter, this is an inductive scenario.

Transductive node classification is a widely studied problem in the GNN community. It has been employed in well-known GNN models like GCNs and GATs and is used in popular benchmark datasets such as Cora, PubMed, and CiteSeer. This setting also has numerous practical applications, including document classification and fraud detection.

2.3 Graph Neural Networks

GNNs are a prevalent method for solving transductive node classification problems. We adopt the message-passing framework for GNNs. A message-passing GNN can be defined as follows:

$h_v^{(0)} = x_v \ (\forall v \in V)$,

$h_v^{(l)} = f_{agg}^{(l)}(h_v^{(l-1)}, \{h_u^{(l-1)} | u \in N(v)\}) \ (\forall l \in [L], v \in V)$,

$\hat{y}_v = f_{pred}(h_v^{(L)}) \ (\forall v \in V)$,

where $f_{agg}^{(l)}$ is the aggregation function at layer l, and f_{pred} is the prediction head, typically implemented using neural networks.

3 LaF is Admissible, but Not Explored Well

We remind the reader of the transductive node classification problem setup. We are given the node labels y_v of the training nodes. A standard approach is to input the node features x_v of a training node v into the model, predict its label, calculate the loss based on the true label y_v, and update the model parameters. However, the use of y_v is not restricted to this. We can also incorporate y_v as a feature for node v. This is the core concept behind LaF.

GNNs with LaF initialize node embeddings as:

$h_v^{(0)} = [x_v; \tilde{y}_v] \in \mathbb{R}^{d+1+|Y|}$,

where $[\cdot;\cdot]$ denotes vector concatenation, and

$\tilde{y}_v = \{ [\ 1; y_v](v \in V_{train})$
$0^{1+|Y|}(v \in V_{test})$,

is the label vector for node v, and 0^d is a zero vector of dimension d. LaF allows GNNs to utilize label information, such as the class distribution in neighboring nodes, to compute node embeddings. These embeddings are likely to be more informative than those without label information. LaF is considered admissible because it only uses information available in the transductive setting.

We emphasize that LaF has not been thoroughly investigated in the GNN literature, despite its simplicity, with a few exceptions. For instance, GCNs and GATs use the transductive setting and could potentially use label information as features. However, they initialize node embeddings as $h_v^{(0)} = x_v$ without using label information. One of the contributions of this paper is to highlight that LaF is permissible in the transductive setting.

Care must be taken when training GNNs with LaF. LaF might negatively impact generalization by creating a shortcut where the model simply copies the label feature $h^{(0)}_{v,d+1:}$ to the prediction. To avoid this, we should remove the labels of the center nodes in the minibatch and treat them as test nodes. Specifically, if $B \subset V_{train}$ is the set of nodes in the minibatch, we set

$\tilde{y}_v = \{ [\ 1; y_v](v \in V_{train} \setminus B)$
$0^{1+|Y|}(v \in V_{test} \cup B)$,

and predict the label $\hat{y}_v$ for $v \in B$, calculating the loss based on $\hat{y}_v$ and y_v. This simulates the transductive setting where the label information of test nodes is unavailable, and GNNs learn to predict test node labels based on the label information and node features of surrounding nodes.

4 LaF Strengthens the Expressive Power of GNNs

We demonstrate that incorporating labels as features (LaF) provably enhances the expressive capabilities of Graph Neural Networks (GNNs). Specifically, we show that GNNs utilizing LaF can effectively represent the label propagation algorithm, a crucial method for transductive node classification, whereas GNNs without LaF cannot achieve this. This finding is significant in its own right and provides a strong motivation for the design of TFGNNs.

Label propagation is a well-established method for transductive node classification. It operates by initiating random walks from a test node and generating the label distribution of the labeled nodes that these random walks encounter first. The following theorem establishes that GNNs with LaF can effectively approximate label propagation.

Theorem 4.1. GNNs with LaF can approximate label propagation with arbitrary precision. Specifically, there exists a series of GNNs $\{f^{(l)}_{agg}\}_l$ and f_{pred} such that for any positive ϵ, for any connected graph $G = (V, E, X)$, for any labeled nodes $V_{train} \subset V$ and node labels $Y_{train} \in Y^{V_{train}}$, and test node $v \in V \setminus V_{train}$, there exists $L \in \mathbb{Z}^+$ such that the $l(\geq L)$-th GNN $(f^{(1)}_{agg}, ..., f^{(l)}_{agg}, f_{pred})$ with LaF outputs an approximation of label propagation with an error of at most ϵ, i.e.,

$||\hat{y}_v - \hat{y}^{LP}_v||_1 < \epsilon$,

where $\hat{y}^{LP}_v$ is the output of label propagation for test node v.

Proof. We prove the theorem by construction. Let

$p_{l,v,i} def=$ Pr[The random walk from node v hits V_{train} within l steps and the first hit label is i].

For labeled nodes, this is a constant:

$p_{l,v,i} = 1_{[i=y_v]}\ (\forall l \in \mathbb{Z}_{\geq 0}, v \in V_{train}, i \in Y)$.

For other nodes, it can be recursively computed as:

$p_{0,v,i} = 0\ (\forall v \in V \setminus V_{train}, i \in Y)$,

$p_{l,v,i} = \sum_{u \in N(v)} \frac{1}{deg(v)} \cdot p_{l-1,u,i}$.

These equations can be represented by GNNs with LaF. The base case

$p_{0,v,i} = \{\, 1_{[i=y_v]}\, (v \in V_{train})$
$0(v \in V \setminus V_{train})$,

can be computed from $\tilde{y}_v$ in $h^{(0)}_v$. Let $f^{(l)}_{agg}$ always concatenate its first argument ($h^{(l-1)}_v$) to the output so the GNN retains input information. $f^{(l)}_{agg}$ handles two cases based on $\tilde{y}_{v,1} \in \{0, 1\}$, indicating whether v is in V_{train}. If $v \in V_{train}$, $f^{(l)}_{agg}$ outputs $1_{[i=y_v]}$, computable from $\tilde{y}_v$ in $h^{(l-1)}_v$. If $v \notin V_{train}$, $f^{(l)}_{agg}$ aggregates $p_{l-1,u,i}$ from $u \in N(v)$ and averages them, as in the recursive equation, realizable by message passing in the second argument of $f^{(l)}_{agg}$.

The final output of the GNN is $p_{l,v,i}$. The output of label propagation can be decomposed as:

$\hat{y}^{LP}_{v,i} =$ Pr[The first hit label is i]

$=$ $p_{l,v,i}+$ Pr[The random walk from node v does not hit V_{train} within l steps and the first hit label is i].

As the second term converges to zero as l increases, GNNs can approximate label propagation with arbitrary precision by increasing l.

We then show that GNNs without LaF cannot represent label propagation.

Proposition 4.2. GNNs without LaF cannot approximate label propagation. Specifically, for any series of GNNs $\{f^{(l)}_{agg}\}_l$ and f_{pred}, there exists a positive ϵ, a connected graph $G = (V, E, X)$, labeled nodes $V_{train} \subset V$, node labels $Y_{train} \in Y^{V_{train}}$, and a test node $v \in V \setminus V_{train}$, such that for any l, the GNN $(f^{(1)}_{agg}, ..., f^{(l)}_{agg}, f_{pred})$ without LaF has an error of at least ϵ, i.e.,

$||\hat{y}_v - \hat{y}_v^{LP}||_1 > \epsilon$,

where $\hat{y}_v^{LP}$ is the output of label propagation for test node v.

Proof. We construct a counterexample. Let G be a cycle of four nodes numbered 1, 2, 3, 4 clockwise. All nodes have the same feature x. Let $V_{train} = \{1, 2\}$ and $Y_{train} = [1, 0]^T$. Label propagation classifies node 4 as class 1 and node 3 as class 0. However, GNNs without LaF always predict the same label for nodes 3 and 4 since they are isomorphic. Thus, for any GNN without LaF, there is an irreducible error for either node 3 or 4.

Theorem 4.1 and Proposition 4.2 demonstrate that LaF provably enhances the expressive power of GNNs. These results indicate that GNNs with LaF are more powerful than traditional message-passing GNNs like GCNs, GATs, and GINs without LaF. Notably, while GINs are considered the most expressive message-passing GNNs, they cannot represent label propagation without LaF, whereas message-passing GNNs with LaF can. This does not lead to a contradiction since the original GINs do not take the label information as input. In other words, the input domains of the functions differ. These findings highlight the importance of considering both the input and the architecture of GNNs to maximize their expressive power.

5 Training-free Graph Neural Networks

We propose training-free graph neural networks (TFGNNs) based on the analysis in the previous section. TFGNNs can be used without training and can also be improved with optional training.

First, we define training-free models.

Definition 5.1 (Training-free Model). We say a parametric model is training-free if it can be used without optimizing the parameters.

It should be noted that nonparametric models are training-free by definition. The real worth of TFGNNs is that it is training-free while it can be improved with optional training. Users can enjoy the best of both worlds of parametric and nonparametric models by choosing the trade-off based on the computational resources for training and the accuracy required.

The core idea of TFGNNs is to embed label propagation in GNNs by Theorem 4.1. TFGNNs are defined as follows:

$h_v^{(0)} = [x_v; \tilde{y}_v]$,

$h_v^{(l)} = \{\ ReLU(S^{(l)} h_v^{(l-1)} + \frac{1}{|N(v)|} \sum_{u \in N(v)} W^{(l)} h_u^{(l-1)}) (v \in V_{train}, l \in [L])$
$ReLU(T^{(l)} h_v^{(l-1)} + \frac{1}{|N(v)|} \sum_{u \in N(v)} W^{(l)} h_u^{(l-1)}) (v \in V_{test}, l \in [L])$,

$\hat{y}_v = softmax(U h_v^{(L)})$,

The architecture of TFGNNs is standard, i.e., TFGNNs transform the center nodes and carry out mean aggregation from the neighboring nodes. The key to TFGNNs lies in initialization. The parameters are initialized as follows:

$S^{(l)}_{-(1+|Y|):,:-(1+|Y|)} = 0,\ S^{(l)}_{-(1+|Y|):,-(1+|Y|):} = I_{1+|Y|},\ V^{(l)}_{-(1+|Y|):} = 0,\ T^{(l)}_{-(1+|Y|):} = 0,\ W^{(l)}_{-(1+|Y|):,:-(1+|Y|)} = 0,$
$W^{(l)}_{-(1+|Y|):,-(1+|Y|):} = I_{1+|Y|}, U_{:,:-|Y|} = 0, U_{:,-|Y|:} = I_{|Y|}$,

i.e., the parameters of the last $(1 + |Y|)$ rows or $|Y|$ rows are initialized by 0 or 1 in a special pattern (Figure 1). Other parameters are initialized randomly, e.g., by Xavier initialization. The following proposition shows that the initialized TFGNNs approximate label propagation.

Proposition 5.2. The initialized TFGNNs approximate label propagation. Specifically,

$h^{(L)}_{v,-(|Y|-i+1)} = p_{L,v,i}$

holds, where $p_{L,v,i}$ is defined in Eq. (8), and

$argmax_i \hat{y}_{v,i} = argmax_i p_{L,v,i}$

holds, and $p_{L,v,i} \to \hat{y}^{LP}_{v,i}$ as $L \to \infty$.

Proof. By the definitions of TFGNNs,

$h^{(0)}_{v,-|Y|:} = \{\ y_v\ (v \in V_{train})$
$0^{|Y|} (v \in V_{test})$,

$h^{(l)}_{v,-|Y|:} = \{\ h^{(l-1)}_{v,-|Y|:} (v \in V_{train}, l \in [L])$
$\frac{1}{|N(v)|} \sum_{u \in N(v)} h^{(l-1)}_{u,-|Y|:} (v \in V_{test}, l \in [L])$.

This recursion is the same as Eqs. (9) – (13). Therefore,

$h^{(L)}_{v,-(|Y|-i+1)} = p_{L,v,i}$

holds. As U picks the last $|Y|$ dimensions, and softmax is monotone,

$argmax_i \hat{y}_{v,i} = argmax_i p_{L,v,i}$

holds. $p_{L,v,i} \to \hat{y}^{LP}_{v,i}$ as $L \to \infty$ is shown in the proof of Theorem 4.1.

Therefore, the initialized TFGNNs can be used for transductive node classification as are without training. The approximation algorithm of label propagation is seamlessly embedded in the model parameters, and TFGNNs can also be trained as usual GNNs.

6 Experiments

6.1 Experimental Setup

We use the Planetoid datasets (Cora, CiteSeer, PubMed), Coauthor datasets, and Amazon datasets in the experiments. We use 20 nodes per class for training, 500 nodes for validation, and the rest for testing in the Planetoid datasets following standard practice, and use 20 nodes per class for training, 30 nodes per class for validation, and the rest for testing in the Coauthor and Amazon datasets. We use GCNs and GATs for the baselines. We use three-layered models with a hidden dimension of 32 unless otherwise specified. We train all models with AdamW with a learning rate of 0.0001 and weight decay of 0.01.

6.2 TFGNNs Outperform Existing GNNs in Training-free Setting

We compare the performance of TFGNNs with GCNs and GATs in the training-free setting by assessing the accuracy of the models when the parameters are initialized. The results are shown in Table 1. TFGNNs outperform GCNs and GATs in all the datasets. Specifically, both GCNs and GATs are almost random in the training-free setting, while TFGNNs achieve non-trivial accuracy. These results validate that TFGNNs meet the definition of training-free models. We can also observe that GCNs, GATs, and TFGNNs do not benefit from LaF in the training-free settings if randomly initialized. These results indicate that both LaF and the initialization of TFGNNs are important for training-free performance.

Table 1: Node classification accuracy in the training-free setting. The best results are shown in bold. CS: Coauthor CS, Physics: Coauthor Physics, Computers: Amazon Computers, Photo: Amazon Photo. TFGNNs outperform GCNs and GATs in all the datasets. These results indicate that TFGNNs are training-free. Note that we use three-layered TFGNNs to make the comparison fair although deeper TFGNNs perform better in the training-free setting as we confirm in Section 6.3.

	Cora	CiteSeer	PubMed	CS	Physics	Computers
GCNs	0.163	0.167	0.180	0.079	0.101	0.023
GCNs + LaF	0.119	0.159	0.407	0.080	0.146	0.061
GATs	0.177	0.229	0.180	0.040	0.163	0.058
GATs + LaF	0.319	0.077	0.180	0.076	0.079	0.025
TFGNNs + random initialization	0.149	0.177	0.180	0.023	0.166	0.158
TFGNNs (proposed)	**0.600**	**0.362**	**0.413**	**0.601**	**0.717**	**0.730**

6.3 Deep TFGNNs Perform Better in Training-free Setting

We confirm that deeper TFGNNs perform better in the training-free setting. We have used three-layered TFGNNs so far to make the comparison fair with existing GNNs. Proposition 5.2 shows that the initialized TFGNNs converge to label propagation as the depth goes to infinity, and we expect that deeper TFGNNs perform better in the training-free setting. Figure 2 shows the accuracy of TFGNNs with different depths for the Cora dataset. We can observe that deeper TFGNNs perform better in the training-free setting until the depth reaches around 10, where the performance saturates. It is noteworthy that GNNs have been known to suffer from the oversmoothing problem, and the performance of GNNs degrades as the depth increases. It is interesting that TFGNNs do not suffer from the oversmoothing problem in the training-free setting. It should be noted that it does not necessarily mean that deeper models perform better in the optional training mode because the optional training may break the structure introduced by the initialization of TFGNNs and may lead to oversmoothing and/or overfitting. We leave it as a future work to overcome these problems by adopting countermeasures such as initial residual and identity mapping, MADReg, and DropEdge.

6.4 TFGNNs Converge Fast

In the following, we investigate the optional training mode of TFGNNs. We train the models with three random seeds and report the average accuracy and standard deviation. We use baseline GCNs without LaF (i.e., the original GCNs) as the baseline.

First, we confirm that TFGNNs in the optional training mode converge faster than GCNs. We show the training curves of TFGNNs and GCNs for the Cora dataset in Figure 3. TFGNNs converge much faster than GCNs. We hypothesize that TFGNNs converge

faster because the initialized TFGNNs are in a good starting point, while GCNs start from a completely random point and require many iterations to reach a good point. We can also observe that fully trained TFGNNs perform on par with GCNs. These results indicate that TFGNNs enjoy the best of both worlds: TFGNNs perform well without training and can be trained faster with optional training.

6.5 TFGNNs are Robust to Feature Noise

As TFGNNs use both node features and label information while traditional GNNs rely only on node features, we expect that TFGNNs are more robust to feature noise than traditional GNNs. We confirm this in this section. We add i.i.d. Gaussian noise with standard deviation σ to the node features and evaluate the accuracy of the models. We train TFGNNs and GCNs with the Cora dataset. The results are shown in Figure 4. TFGNNs are more robust to feature noise especially in high noise regimes where the performance of GCNs degrades significantly. These results indicate that TFGNNs are more robust to i.i.d. Gaussian noise to the node features than traditional GNNs.

7 Related Work

7.1 Labels as Features and Training-free GNNs

The most relevant work is by Wang et al., who proposed to use node labels in GNNs. This technique was also used by Addanki et al. and analyzed by Wang et al. The underlying idea is common with LaF, i.e., use of label information as input to transductive GNNs. A similar result as Theorem 4.1 was also shown in Wang et al. However, the focus is different, and there are different points between this work and theirs. We propose the training-free + optional training framework for the first time. The notable characteristics of GNNs are (i) TFGNNs receive both original features and LaF, (ii) TFGNNs can be deployed without training, and (iii) TFGNNs can be improved with optional training. Besides, we provide detailed analysis and experiments including the speed of convergence and noise robustness. Our results provide complementary insights to the existing works.

Another related topic is graph echo state networks, which lead to lightweight models for graph data. The key idea is to use randomly initialized fixed weights for aggregation. The main difference is that graph echo state networks still require to train the output layer, while TFGNNs can be used without training. These methods are orthogonal, and it is an interesting direction to combine them to further improve the performance.

7.2 Speeding up GNNs

Various methods have been proposed to speed up GNNs to handle large graph data. GraphSAGE is one of the earliest methods to speed up GNNs. GraphSAGE employs neighbor sampling to reduce the computational cost of training and inference. It samples a fixed number of neighbors for each node and aggregates the features of the sampled neighbors. An alternative sampling method is layer-wise sampling introduced in FastGCN. Huang et al. further improved FastGCN by using an adaptive node sampling technique to reduce the variance of estimators. LADIES combined neighbor sampling and layer-wise sampling to take the best of both worlds. Another approach is to use smaller training graphs. ClusterGCN uses a cluster of nodes as a mini-batch. GraphSAINT samples subgraphs by random walks for each mini-batch.

It should also be noted that general techniques to speed up neural networks, such as mixed-precision training, quantization, and pruning can be applied to GNNs.

These methods mitigate the training cost of GNNs, but they still require many training iterations. In this paper, we propose training-free GNNs, which can be deployed instantly as soon as the model is initialized. Besides, our method can be improved with optional training. In the optional training mode, the speed up techniques mentioned above can be combined with our method to reduce the training time further.

7.3 Expressive Power of GNNs

Expressive power (or representation power) means what kind of functional classes a model family can realize. The expressive power of GNNs is an important field of research in its own right. If GNNs cannot represent the true function, we cannot expect GNNs to work well however we train them. Therefore, it is important to elucidate the expressive power of GNNs. Originally, Morris et al. and Xu et al. showed that message-passing GNNs are at most as powerful as the 1-WL test, and they proposed k-GNNs and GINs, which are as powerful as the k-(set)WL and 1-WL tests, respectively. GINs are the most powerful message-passing GNNs. Sato and Loukas showed that message-passing GNNs are as powerful as a computational model of distributed local algorithms, and they proposed GNNs that are as powerful as port-numbering and randomized local algorithms. Loukas showed that GNNs are Turing-complete under certain conditions (i.e., with unique node ids and infinitely increasing depths). Some other works showed that GNNs can solve or cannot solve some specific problems, e.g., GNNs can recover the underlying geometry, GNNs cannot recognize bridges and articulation points. There are various efforts to improve the expressive power of GNNs by non-message-passing architectures. We refer the readers to survey papers for more details on the expressive power of GNNs.

We contributed to the field of the expressive power of GNNs by showing that GNNs with LaF are more powerful than GNNs without LaF. Specifically, we showed that GNNs with LaF can represent an important model, label propagation, while GNNs without LaF cannot. It should be emphasized that GINs, the most powerful message-passing GNNs, and Turing-complete GNNs cannot represent label propagation without LaF because they do not have access to the label information label propagation uses, and also noted that GINs traditionally do not use LaF. This result indicates that it is important to consider what to input to the GNNs as well as the architecture of the GNNs for the expressive power of GNNs. This result provides a new insight into the field of the expressive power of GNNs.

8 Limitations

Our work has several limitations. First, LaF and TFGNNs cannot be applied to inductive settings while most GNNs can. We do not regard this as a negative point. Popular GNNs such as GCNs and GATs are applicable to both transductive and inductive settings and are often used for transductive settings. However, this also means that they do not take advantage of transductive-specific structures (those that are not present in inductive settings). We believe that it is important to exploit inductive-specific techniques for inductive settings and transductive-specific techniques (such as LaF) for transductive settings in order to pursue maximum performance.

Second, TFGNNs cannot be applied to heterophilious graphs, or its performance degrades as TFGNNs are based on label propagation. The same argument mentioned above applies. Relying on homophilious graphs is not a negative point in pursuing maximum performance. It should be noted that LaF may also be exploited in heterophilious settings as well. Developing training-free GNNs for heterophilious graphs based on LaF is an interesting future work.

Third, we did not aim to achieve the state-of-the-art performance. Exploring the combination of LaF with fancy techniques to achieve state-of-the-art performance is left as future work.

Finally, we did not explore applications of LaF other than TFGNNs. LaF can help other GNNs in non-training-free settings as well. Exploring the application of LaF to other GNNs is left as future work.

9 Conclusion

In this paper, we made the following contributions.

* We advocated the use of LaF in transductive learning (Section 3). * We confirmed that LaF is admissible in transductive learning, but LaF has not been explored in the field of GNNs such as GCNs and GATs. * We formally showed that LaF strengthens the expressive power of GNNs (Section 4). * We showed that GNNs with LaF can represent label propagation (Theorem 4.1) while GNNs without LaF cannot (Proposition 4.2). * We proposed training-free graph neural networks, TFGNNs (Section 5). * We showed that TFGNNs defined by Eqs. (19) – (29) meet the requirementsarticle graphicx