MOTIF-INDUCED GRAPH NORMALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

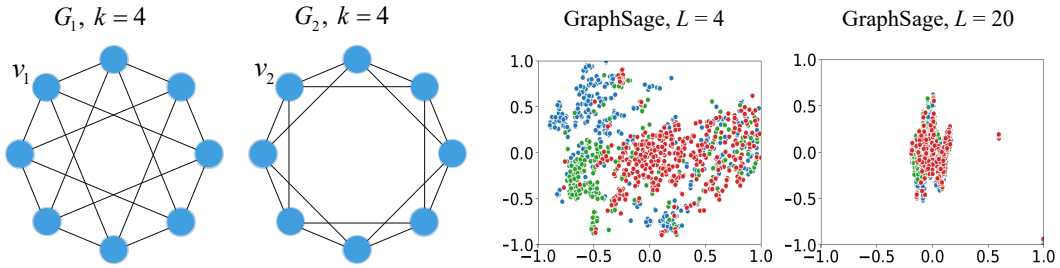
Graph Neural Networks (GNNs) have emerged as a powerful category of learning architecture for handling graph-structured data in the non-Euclidean domain. Despite their success, existing GNNs typically suffer from the insufficient expressive power bottlenecked by Weisfeiler-Lehman (WL) test, and meanwhile are prone to the over-smoothing situation with increasing layer numbers. In this paper, we strive to strengthen the discriminative capabilities of GNNs by devising a dedicated plug-and-play normalization scheme, termed as *Motif-induced Normalization* (MotifNorm), that explicitly considers the intra-connection information within each node-induced subgraph. To this end, we embed the motif-induced structural weights at the beginning and the end of the standard BatchNorm, as well as incorporate the graph instance-specific statistics for improved distinguishable capabilities. In the meantime, we provide the theoretical analysis to support that, with the proposed elaborated MotifNorm, an arbitrary GNNs is capable of more expressive abilities than the 1-WL test in distinguishing k-regular graphs. Furthermore, the proposed MotifNorm scheme is also exemplified to be able to alleviate the over-smoothing phenomenon. Experimental results on ten popular benchmarks across all the tasks of the graph-, node-, as well as link-level property predictions, demonstrate the effectiveness of the proposed method. Our code is made available in the supplementary material.

1 INTRODUCTION

In recent years, Graph Neural Networks (GNNs) (Defferrard et al., 2016; Kipf & Welling, 2017; Xu et al., 2019) have emerged as the mainstream deep learning architectures to analyze irregularly structured samples where information is present in the form of graphs, which encode the complex connection among various instances. As a powerful class of graph-relevant deep networks, these architectures have shown encouraging performance in various domains such as cell clustering (Li et al., 2022; Alghamdi et al., 2021), chemical prediction (Tavakoli et al., 2022; Zhong et al., 2022), social networks (Bouritsas et al., 2022; Dwivedi et al., 2022b), traffic networks (Bui et al., 2021; Li & Zhu, 2021), combinatorial optimization (Schuetz et al., 2022; Cappart et al., 2021), and power grids (Boyaci et al., 2021; Chen et al., 2022a).

Most existing GNNs employ the message-passing aggregation mechanism over multisets to optimize the node features from local neighborhood representations (Kipf & Welling, 2017; Gilmer et al., 2017; Veličković et al., 2018; Xu et al., 2019; Yang et al., 2020b; Hao et al., 2021; Dwivedi et al., 2022b). However, such straightforward neighborhood aggregation scheme leads to the limitation that the expressive capability of GNNs is at most as powerful as the Weisfeiler-Leman (WL) (Weisfeiler & Leman, 1968) isomorphism test for graph classification (Xu et al., 2019), meaning that simple graph structures are still indistinguishable. An example is shown in Figure 1 (a), where it is still challenging for existing GNNs to separate the two k-regular graphs G_1 and G_2. Furthermore, the advances in recent researches (Li et al., 2018; Zhao & Akoglu, 2020) show that the formulation of GNNs is a particular format of Laplacian smoothing, which thereby results in the over-smoothing issue, especially when GNNs go deeper, as shown in Figure 1 (b). As a result of increasing layer numbers, GNNs are typically prone to indistinguishable node representations for node and link predictions (Gupta et al., 2021; Chen et al., 2022b).

In this paper, we strive to develop a general framework to improve the graph expressivity over the prevalent message-passing GNNs in distinguishing graph structures and alleviate the over-smoothing dilemma for node-based predictions. Driven by the fact that deep models usually follow the CNA



(a) Two k-regular graphs (b) Over-smoothing phenomenon

Figure 1: (a) The illustration of two different k-regular graphs, where $k = 4$ in G_1 and G_2. (b) The t-SNE illustration of over-smoothing issue on `Cora` dataset (show the first three categories).

architecture, *i.e.*, a stack of convolution, normalization and activation, where the normalization module generally follows `GNNs` convolution operations, we accordingly focus on developing a higher-expressivity generalized normalization scheme to enhance the discriminative abilities of various `GNNs` architectures. Despite many dedicated graph normalization methods (Zhao & Akoglu, 2020; Zhou et al., 2021; Yang et al., 2020a; Cai et al., 2021; Dwivedi et al., 2022a), these approaches are usually task-specific and sophisticated in `GNNs`' architectures, which means that they are not always significant nor consistent in general contribution to extending `GNNs`' expressivity in distinguishing graphs and alleviating the over-smoothing phenomenon.

This paper addresses this challenge by devising an innovative normalization mechanism, termed as *Motif-induced Normalization* (`MotifNorm`), that explicitly considers the intra-connection information in each node-induced subgraph (*i.e.*, motif (Leskovec, 2021)) and embeds the achieved motif-induced structural factors into the normalization module to improve the expressive power of `GNNs`. Specifically, we start by empirically disentangling the standard normalization into two stages, *i.e.*, centering & scaling (`CS`) and affine transformation (`AT`) operations. We then concentrate on mining the intra-connection information in the induced subgraph, and develop two elaborated strategies, termed as *representation calibration* and *representation enhancement*, to embed the achieved structural information into `CS` and `AT` operations. Eventually, we theoretically demonstrate that `MotifNorm` can extend `GNNs` beyond the `1-WL` in distinguishing k-regular graphs and help alleviate the over-smooting issue.

In sum, the contributions of this work can be summarized as follows:

- Driven by the conjecture that a higher-expressivity normalization with abundant graph structure power can generally strengthen the `GNNs`' performance, we develop a novel normalization scheme, termed as `MotifNorm`, to embed structural information into node representations.
- We develop two elaborated strategies, *i.e.*, representation calibration and representation enhancement, tailored to embed the *motif-induced* structural factor into `CS` and `AT` operations for the establishment of `MotifNorm` in `GNNs`.
- We provide the theoretical analysis to support the claim that `MotifNorm` can extend `GNNs` beyond the `1-WL` test in distinguishing k-regular graphs and also exemplify why it can help alleviate the over-smoothing issue when `GNNs` go deeper.

Extensive experiments on ten popular benchmarks across various domains, such as graph, node and link property predictions, demonstrate that the proposed model is efficient and can consistently yield encouraging results, and meanwhile maintaining the computational simplicity.

2 PRELIMINARY

In this section, we begin by understanding the issue of the graph isomorphism test and the over-smoothing phenomenon in `GNNs`, along the way, introduce notations in this paper. Let $G = (V_G, E_G)$ denotes a undirected graph with n vertices and m edges, where $V_G = \{v_1, v_2, ..v_n\}$ is an unordered set of vertices and $E_G \subseteq V_G \times V_G$ is a set of edges. $\mathcal{N}(v) = \{u \in V_G | (v, u) \in E_G\}$ denotes the neighbor set of vertex v, and its neighborhood subgraph S_v is induced by $\tilde{\mathcal{N}}(v) = \mathcal{N}(v) \cup v$, which

contains all edges in E_G that have both endpoints in $\tilde{\mathcal{N}}(v)$. As shown in Figure 2, S_{v_1} and S_{v_2} are the induced subgraphs of v_1 and v_2 in Figure 1 (a). The feature matrix $\mathrm{H} = [h_1, h_2, ..., h_n] \in \mathbb{R}^{n \times d}$ is the learned feature from GNNs, where d is the embedded dimension.

2.1 Graph Isomorphism Issue

The graph isomorphism issue is a challenging problem of determining whether two finite graphs are topologically identical. The WL algorithm (Weisfeiler & Leman, 1968) is a well-established framework that distinguishes a broad class of graphs. In detail, 1-WL refines color by iteratively aggregating and hashing the multiset into unique labels:

$$h_v^{(t)} = \texttt{Hash}(h_v^{t-1}, \mathcal{A}\{h_u^{t-1} | u \in \mathcal{N}(v)\}), \tag{1}$$

where $h_v^{(t)}$ is the t-th layer representation of node v. $\mathcal{A}$ is the aggregating function to aggregate its neighbors' labels, and Hash is the injective mapping function to get a unique new label. However, **1-WL can not distinguish subtree-isomorphic subgraphs**, *e.g.*, S_{v_1}, S_{v_2} in Figure 2, which are defined as follow:

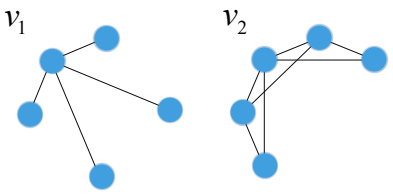


Figure 2: The induced subgraphs of v_1 and v_2 from G_1 and G_2 in Figure 1 (a).

Definition 1. *S_{v_i} and S_{v_j} are subtree-isomorphic, $S_{v_i} \simeq_{subtree} S_{v_j}$, if there exists a bijective mapping $f : \tilde{\mathcal{N}}(v_i) \to \tilde{\mathcal{N}}(v_j)$ such that $f(v_i) = v_j$ and for any $v' \in \tilde{\mathcal{N}}(v_i)$ and $f(v') = u', h_{v'} = h_{u'}$.*

Expressivities of GNNs. GNNs' expressivities are as powerful as the 1-WL in distinguishing non-isomorphic graphs, *w.r.t.* subtree-isomorphic at each layer, which is stated in the following theorem.

Theorem 1. *GNNs are as powerful the as 1-WL test in distinguishing non-isomorphic graphs while any two different subgraphs S_{v_i}, S_{v_j} are subtree-isomorphic (i.e., $S_{v_i} \simeq_{subtree} S_{v_j}$), or GNNs can map two different subgraphs into two different embeddings if and only if $S_{v_i} \not\simeq_{subtree} S_{v_j}$.*

The complete proof is provided in Appendix A.1.

2.2 Over-smoothing issue

For the GNNs convolution, a graph structure G can be equivalently defined as an adjacency matrix $\mathrm{A} \in \mathbb{R}^{n \times n}$ to update features with the degree matrix $\mathrm{D} = \mathrm{diag}(\deg_1, \deg_2, ..., \deg_n) \in \mathbb{R}^{n \times n}$. Let $\tilde{\mathrm{A}} = \mathrm{A} + \mathrm{I}$ and $\tilde{\mathrm{D}} = \mathrm{D} + \mathrm{I}$ denote the augmented adjacency and degree matrices by adding self-loops. $\tilde{\mathrm{A}}_{\mathtt{sym}} = \tilde{\mathrm{D}}^{-1/2}\tilde{\mathrm{A}}\tilde{\mathrm{D}}^{-1/2}$ and $\tilde{\mathrm{A}}_{\mathtt{rw}} = \tilde{\mathrm{D}}^{-1}\tilde{\mathrm{A}}$ represent symmetrically and nonsymmetrically normalized adjacency matrices, respectively. Given the input node feature matrix H^{t-1}, the basic format of GNNs convolution follows:

$$\mathrm{H}^t = (\mathrm{I} - \alpha\mathrm{I})\mathrm{H}^{t-1} + \alpha\tilde{\mathrm{A}}_{\mathtt{rw}}\mathrm{H}^{t-1}, \tag{2}$$

where graph convolution usually lets $\alpha = 1$ and uses the symmetrically normalized Laplacian to obtain $\mathrm{H}^t = \tilde{\mathrm{A}}_{\mathtt{sym}}\mathrm{H}^{t-1}$. To analyze GNNs' working mechanism, Li et al. (2018) showed that the graph convolution is a particular form of Laplacian smoothing, which makes the node representation from different clusters become mixed up. Furthermore, Zhao & Akoglu (2020) discovered that Laplacian smoothing washes away the information from all the features, *i.e.* the representations with regard to different columns of the feature matrix, and thus makes nodes indistinguishable.

Representatives of GNNs. As a natural result of GNNs going deeper, node representations become similar, and signals are washed away. Therefore, sample representations from different clusters become indistinguishable so that GNNs' performance suffers from this phenomenon for node-relevant tasks, *e.g.*, node and link classification (Gupta et al., 2021; Chen et al., 2022b).

3 Motif-induced Graph Normalization

In this section, we present a generalized graph normalization framework, termed as MotifNorm, which enables us to inject substructure weight into GNNs-based architectures. ***Notably***, the designed weight factor needs at least to distinguish two subgraphs S_{v_1}, S_{v_2} shown in Figure 2, where two subgraphs contain the same number of nodes. Thus, we comprehensively take the number of edges and the number of nodes into consideration, and give the definition of *motif-induced* weight.

Definition 2. *The motif-induced information $\xi(S_{v_i})$ denotes the structural weight of substructure S_{v_i}, i.e., node-induced subgraph (Leskovec, 2021) with regard to v_i. To exhibit the structure-specific information of different subgraphs, we give the formulation of ξ as:*

$$\xi(S_{v_i}) = \varphi(|E_{S_{v_i}}|)\psi(|V_{S_{v_i}}|), \tag{3}$$

where $|\cdot|$ denotes the number of samples in a countable set. $\varphi(|E_{S_{v_i}}|) = 2|E_{S_{v_i}}|/(|V_{S_{v_i}}|\cdot(|V_{S_{v_i}}|-1))$ and $\psi(|V_{S_{v_i}}|) = |V_{S_{v_i}}|^2$ respectively refers to the density and power of S_{v_i}. This definition focuses more on edge information while two different subgraphs are subtree-isomorphic, on the contrary, focuses more on node power. Moreover, it meets the condition of distinguishing two subgraphs in Figure 2, and allows us to design distinctive normalization to generally improve GNNs' expressivity.

3.1 MotifNorm for GNNs

Standard normalization can be empirically divided into two stages, *i.e.*, centering & scaling (CS) and affine transformation (AT) operations. For the input features $\mathrm{H} \in \mathbb{R}^{n\times d}$, the CS and AT follow:

$$\begin{aligned} \mathtt{CS}: \mathrm{H}_{\mathtt{CS}} &= \frac{\mathrm{H} - \mathbb{E}(\mathrm{H})}{\sqrt{\mathbb{D}(\mathrm{H}) + \epsilon}}, \\ \mathtt{AT}: \mathrm{H}_{\mathtt{AT}} &= \mathrm{H}_{\mathtt{CS}} \odot \gamma + \beta, \end{aligned} \tag{4}$$

where $\odot$ is the dot product with the broadcast mechanism. $\mathbb{E}(\mathrm{H})$ and $\mathbb{D}(\mathrm{H})$ denote mean and variance statistics, and γ, $\beta \in \mathbb{R}^{1\times d}$ are the learned scale and bias factors. The main drawback of existent normalizations is the absence of graph structure information, which downgrades the graph expressivity to distinguish, *e.g.*, isomorphic graphs and other graph symmetries. In this work, we aim to embed *motif-induced* weights into BatchNorm, and thus take a batch of graphs for example.

Batch Graphs. For a bacth of graphs $\mathcal{G} = \{G_1, ..., G_m\}$ with node set $V_{\mathcal{G}} = V_{G_1} \cup V_{G_2}, ..., V_{G_m} = \{v_1, ..., v_n\}$ and feature matrix $\mathrm{H} \in \mathbb{R}^{n\times d}$. Motif-induced weights of this batch nodes is represented as $\mathrm{M}_{\mathcal{G}} = \xi(S_{V_{\mathcal{G}}}) = [\xi(S_{V_{G_1}}); \xi(S_{V_{G_2}}); ...; \xi(S_{V_{G_m}})] = [\xi(S_{v_1}), ..., \xi(S_{v_n})] \in \mathbb{R}^{n\times 1}$. The segment summation-normalization $\mathrm{M}_{\mathtt{SN}} = [\mathcal{F}(\xi(S_{V_{G_1}})); \{\mathcal{F}(\xi(S_{V_{G_2}})); ...; \mathcal{F}(\xi(S_{V_{G_m}}))] \in \mathbb{R}^{n\times 1}$ where $\mathcal{F}(\xi(S_{V_{G_i}})) = \xi(S_{V_{G_i}})/\sum \xi(S_{V_{G_i}}) \in \mathbb{R}^{|V_{G_i}|\times 1}$, denotes the summation-normalization operation in each graph. To establish the MotifNorm, we develop two strategies, *i.e.*, representation calibration (RC) and representation enhancement (RE) as follow:

Representation Calibration (RC). The RC operation is added before the CS stage, which means that it calibrates the inputs by injecting the structure information into representations. For the input feture $\mathrm{H} \in \mathbb{R}^{n\times d}$, the RC is formulated as:

$$\mathtt{RC}: \mathrm{H}_{\mathtt{RC}} = \mathrm{H} + w_{\mathtt{RC}} \odot \mathrm{H}_{\mathtt{SA}} \odot \mathrm{M}_{\mathtt{RC}}, \tag{5}$$

where $w_{\mathtt{RC}} \in \mathbb{R}^{1\times d}$ is a learned weight parameter. $\mathrm{H}_{\mathtt{SA}} \in \mathbb{R}^{n\times d}$ is the segment averaging of H, obtained by performing averaging operations within each graph. $\mathrm{M}_{\mathtt{RC}} = \mathrm{M}_{\mathtt{SN}} \cdot \mathrm{M}_{\mathcal{G}} \in \mathbb{R}^{n\times 1}$ is the calibration factor for RC (This specific design is explained in Appendix A.2), where $\cdot$ is the dot product operation. Embedding the motif-induced weights into normalization, along with the mean statistics of each graph, strengthens the structural power in node representations and balances the distribution differences among the batch graphs and graph instance-specific statistics. Furthermore, it is a particular type of graph decentering, which has been proven to accelerate GNNs' training (Cai et al., 2021). The following theorem provides a complete analysis.

Theorem 2. *RC operation calibrates input features for the following CS, considering the motif-induced structural weights and the graph instance-specific statistics, which benefits strengthening the structural power in node representations, along with stabling and accelerating GNNs' training.*

The complete proof is provided in Appendix A.2.

Representation Enhancement (RE). Obtaining $\mathrm{H}_{\mathtt{CS}}$ by performing CS operation after $\mathrm{H}_{\mathtt{RC}}$, we embed motif-induced structural information into AT stage to enhance the representations. The formulation of RE is written as follows:

$$\mathtt{RE}: \mathrm{H}_{\mathtt{RE}} = \mathrm{H}_{\mathtt{CS}} \cdot \mathtt{Pow}(\mathrm{M}_{\mathtt{RE}}, w_{\mathtt{RE}}), \tag{6}$$

where $w_{\mathtt{RE}} \in \mathbb{R}^{1\times d}$ is a learned weight parameter, and $\mathtt{Pow}(\cdot)$ is the exponential function. To imitate affine weights in AT for each channel, we perform the segment summation-normlization on

calibration factor $\mathrm{M}_{\mathtt{RC}}$ and repeats d columns to obtain enhancement factor $\mathrm{M}_{\mathtt{RE}} \in \mathbb{R}^{n \times d}$, which ensures column signatures of $\mathtt{Pow}(\mathrm{M}_{\mathtt{RE}}, w_{\mathtt{RE}}) - 1$ are consistent. Right after CS operation, node features are stricted into a fixed variance range and distinctive information is slightly weakened. In the meantime, the structural information will be washed away and the node representations become indistinguishable with increasing layers. Thus, we design above RE operation to ensure that motif-induced weights is embedded to maintain the structural information in the final representation.

3.2 Implementation and Analysis of MotifNorm

At the implementation stage, we merge RE operation into AT for a simpler formultation description. Given the input feature $\mathrm{H} \in \mathbb{R}^{n \times d}$, the formulation of the MotifNorm is written as:

$$\begin{aligned} \mathtt{RC}: \mathrm{H}_{\mathtt{RC}} &= \mathrm{H} + w_{\mathtt{RC}} \odot \mathrm{H}_{\mathtt{SA}} \odot \mathrm{M}_{\mathtt{RC}}, \\ \mathtt{CS}: \mathrm{H}_{\mathtt{CS}} &= \frac{\mathrm{H}_{\mathtt{RC}} - \mathbb{E}(\mathrm{H}_{\mathtt{RC}})}{\sqrt{\mathbb{D}(\mathrm{H}_{\mathtt{RC}}) + \epsilon}}, \\ \mathtt{AT}: \mathrm{H}_{\mathtt{AT}} &= \mathrm{H}_{\mathtt{CS}} \cdot (\gamma + \mathbb{P})/2 + \beta, \end{aligned} \tag{7}$$

where $\mathbb{P} = \mathtt{Pow}(\mathrm{M}_{\mathtt{RE}}, w_{\mathtt{RE}})$ and $\mathrm{H}_{\mathtt{AT}}$ is the output of MotifNorm. To this end, we add RC and RE operations at the beginning and ending of the original BatchNorm layer to strengthen the expressivity power after GNNs' convolution. To be specific, we provide the following theoretical analysis to show that MotifNorm can extend GNNs' expressivities beyond the 1-WL in distinguishing k-regular graphs and helps alleviate the over-smoothing issue while GNNs go deeper.

Theorem 3. *MotifNorm extends GNNs beyond the 1-WL test in distinguishing k-regular graphs and helps alleviate the over-smoothing issue when GNNs' layers become deeper.*

proof sketch. An arbitrary GNNs equipped with MotifNorm can at least distinguish two subtree-isomorphic structures S_{v_1} and S_{v_2} in Figure 2 and embed substructural weights to scatter the node representations even though different samples become similar. The complete proof of this theorem and detailed expressivity analysis of GNNs are provided in Appendix A.3.

4 Experiments

To demonstrate the effectiveness of the proposed MotifNorm in different GNNs, we conduct experiments on three graph-structured tasks, including graph-, node- and link-level predictions.

Benchmark Datasets. Ten datasets are employed in three types of tasks, including (i) Graph predictions: IMDB-BINARY, ogbg-molhiv, ogbg-moltoxcast, ogbg-ppa and ZINC. (ii) Node predictions: Cora, Pubmed, Citeseer and ogbn-proteins. (iii) Link predictions: ogbl-collab. The basic statistics of these datasets are summarized in Table 1.

Table 1: The statistic information of the benchmark datasets under different graph-structured tasks.

Dataset Name	Task Level	#Graphs	Avg. #Nodes	Avg. #Edges
IMDB-BINARY	Graph	1,000	19.8	193.1
ogbg-moltoxcast	Graph	8,576	18.8	19.3
ogbg-ppa	Graph	158,100	243.4	2,266.1
ogbg-molhiv	Graph	41,127	25.5	27.5
ZINC	Graph	10,000	23.2	49.8
Cora	Node	1	2,708	5,429
Citeseer	Node	1	3,327	4,732
Pubmed	Node	1	19,717	44,338
ogbn-proteins	Node	1	132,534	39,561,252
ogbl-collab	Link	1	235,868	1,285,465

Baseline Methods. We compare the proposed MotifNorm to various types of graph-structured normalization baselines, including BatchNorm (Ioffe & Szegedy, 2015), GraphNorm (Cai et al., 2021), ExpreNorm (Dwivedi et al., 2022a) for graph predictions, and GroupNorm (Zhou et al., 2020), PairNorm (Zhao & Akoglu, 2020), MeanNorm (Yang et al., 2020a), NodeNorm (Zhou et al., 2021) for all three tasks.

More details about the datasets, baselines and experimental setups are provided in Appendix B.1.

In the following experiments, we aim to answer the questions: (i) Can MotifNorm improve the expressivity for graph isomorphism test, especially go beyond 1-WL on k-regular graphs? (Section 4.1) (ii) Can MotifNorm help alleviate the over-smoothing issue? (Section 4.2) (iii) Can MotifNorm generalize to various graph tasks? (Section 4.3)

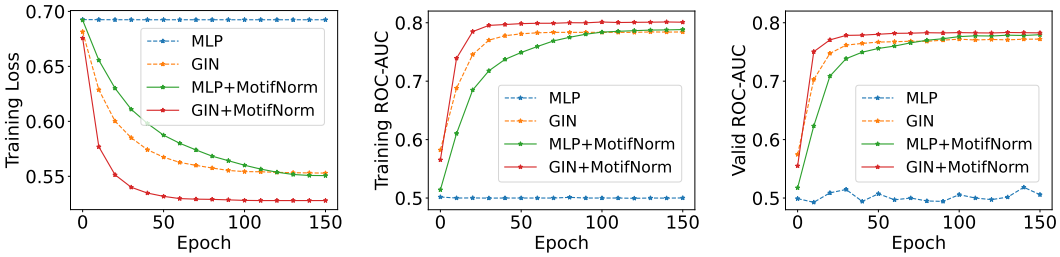


Figure 3: Learning curves of one-layer `MLP`, `GIN`, `MLP` + `MotifNorm` and `GIN` + `MotifNorm` on `IMDB-BINARY` dataset with various k-regular graphs.

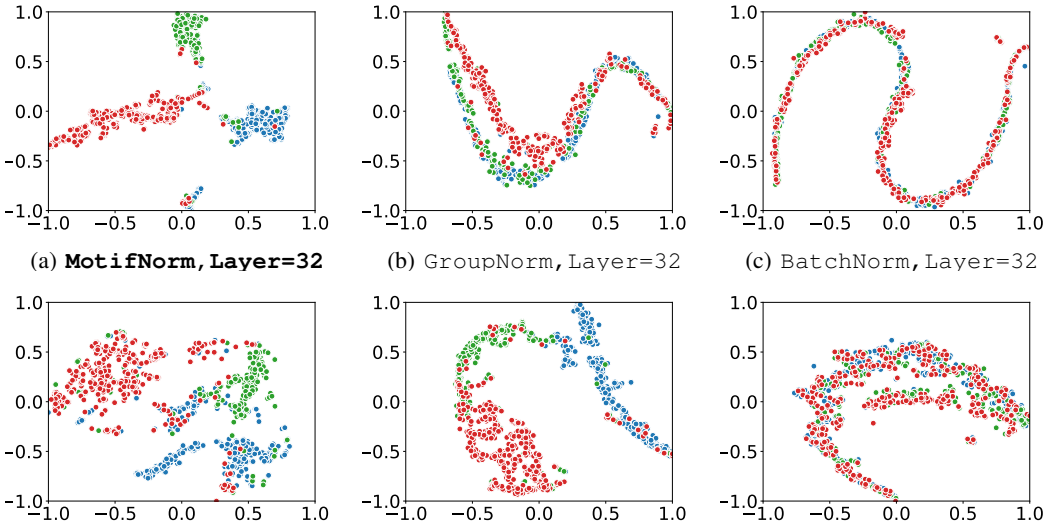
Table 2: Experimental results on `IMDB-BINARY` dataset with various k-regular graphs. The best results under different backbones are highlighted with **boldface.**

	Normalization		Layers	ROC-AUC		LOSS		#Epochs
	`Batchnorm`	`MotifNorm`		Test Split	Valid Split	Test Split	Valid Split	
`MLP`	–	–	1	56.66 ± 1.09	57.05 ± 0.81	0.6961	0.6961	36.40
	✓	–	1	56.86 ± 0.72	57.83 ± 0.84	0.6954	0.6954	112.90
	–	✓	1	$\mathbf{78.16 \pm 0.52}$	$\mathbf{78.69 \pm 0.44}$	0.5631	0.5581	190.70
`GIN`	–	–	1	77.40 ± 0.20	77.69 ± 0.13	0.5684	0.5658	196.60
	✓	–	1	77.71 ± 0.19	78.11 ± 0.10	0.5655	0.5593	152.28
	–	✓	1	$\mathbf{78.42 \pm 0.15}$	$\mathbf{78.89 \pm 0.15}$	0.5584	0.5565	178.20
`GSN`	✓	–	1	77.50 ± 0.18	77.54 ± 0.16	0.5696	0.5688	67.20
`GraphSNN`	✓	–	1	77.52 ± 0.16	77.61 ± 0.18	0.5691	0.5671	89.50
`GIN`	–	–	4	78.22 ± 0.41	78.27 ± 0.45	0.5602	0.5553	154.60
	✓	–	4	78.41 ± 0.21	78.76 ± 0.17	0.5625	0.5590	161.00
	–	✓	4	$\mathbf{79.55 \pm 0.35}$	$\mathbf{79.62 \pm 0.38}$	0.5539	0.5510	133.20
`GCN`	–	–	4	68.99 ± 1.38	69.08 ± 1.81	0.6920	0.6921	33.10
	✓	–	4	76.75 ± 1.31	76.96 ± 0.69	0.5755	0.5640	153.80
	–	✓	4	$\mathbf{78.78 \pm 1.01}$	$\mathbf{79.03 \pm 0.61}$	0.5597	0.5504	61.90
`GAT`	–	–	4	69.07 ± 1.59	69.78 ± 1.65	0.6904	0.6891	33.00
	✓	–	4	75.10 ± 1.51	75.95 ± 1.27	0.5866	0.5852	156.00
	–	✓	4	$\mathbf{78.87 \pm 0.80}$	$\mathbf{79.20 \pm 0.56}$	0.5559	0.5448	122.00
`GSN`	✓	–	4	78.90 ± 0.63	79.28 ± 0.70	0.5555	0.5543	94.20
`GraphSNN`	✓	–	4	79.16 ± 0.67	79.35 ± 0.82	0.5541	0.5530	73.60

4.1 Experimental Analysis on Graph Isomorphism Test

The `IMDB-BINARY` is a well-known graph isomorphism test dataset consisting of various k-regular graphs, which has become a common-used benchmark for evaluating the expressivity of `GNNs`. To obtain the training, valid and test sets that follow the same distribution, we adopt a hierarchical dataset splitting strategy based on the structural statistics of graphs (More detailed description are provided in Appendix B.1). For graph isomorphism test, Graph Isomorphism Network (`GIN`) (Xu et al., 2019) is known to be as powerful as `1-WL`. Notably, `GIN` consists a neighborhood aggregation operation and a multi-layer perception layer (`MLP`), and this motivates a comparison experiment: comparing a one-layer `MLP+MotifNorm` with one-layer `GIN` to directly demonstrate `MotifNorm`'s expressivities in distinguishing k-regular graphs.

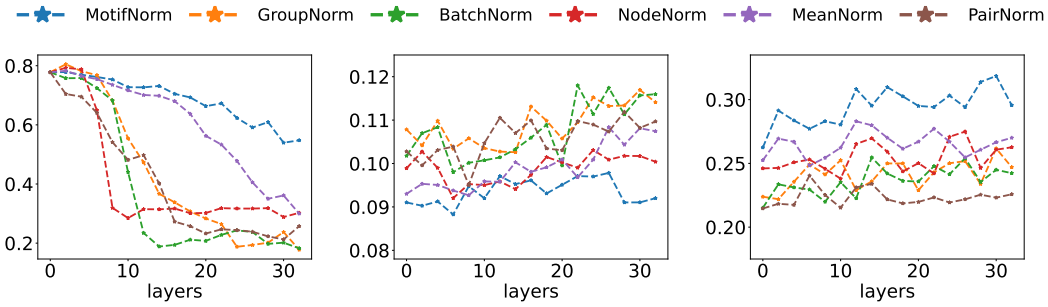
As illustrated in Figure 3, the vanilla `MLP` cannot capture any structural information and perform poorly, while the proposed `MotifNorm` method successfully improve the performance of `MLP` and even exceeds the vanilla `GIN`. Furthermore, Table 2 provides the quantitative comparison results, where `GSN` (Bouritsas et al., 2022) and `GraphSNN` (Wijesinghe & Wang, 2022) are two recent popular methods realizing the higher expressivity than the `1-WL`. From these comparison results, the performance of one-layer `MLP` with `MotifNorm` is better than that of one-layer `GIN`, `GSN`, and `GraphSNN`. Moreover, the commonly used `GNNs` equipped with `MotifNorm`, *e.g.*, `GCN` and `GAT`, achieve higher ROC-AUC than `GIN` when the layer is set as 4. `GIN` with `MotifNorm` achieves better performance and even goes beyond the `GSN` and `GraphSNN`. More detailed results on `IMDB-BINARY` are provided in Appendix B.2.



(a) **MotifNorm, Layer=32** (b) GroupNorm, Layer=32 (c) BatchNorm, Layer=32

(d) NodeNorm, Layer=32 (e) MeanNorm, Layer=32 (f) PairNorm, Layer=32

Figure 4: The t-SNE visualization of node representations using GCN with different normalization methods on Cora dataset.



(a) Accuracy (b) Intra-class distance (c) Inter-class distance

Figure 5: Experimental results of GCN with different normalization methods on Cora dataset.

4.2 Experimental Analysis on the Over-smoothing Issue

To show the effectiveness of MotifNorm for alleviating the over-smoothing issue in GNNs, we visualize the first three categories of Cora dataset in 2D space for a better illustration. We select PairNorm, NodeNorm, MeanNorm, GroupNorm and BatchNorm for comparison and set the number of layer as 32. Figure 4 ((a)∼(f)) show the t-SNE visualization of different normalization techniques, and we can find that none of them suffer from the over-smoothing issue like GraphSage with BatchNorm (shown in Figure 1 (b)). However, MotifNorm can better distinguish different categories into different clusters, *i.e.*, the other normalization methods may lead to the loss of discriminant information and make the representations entangled.

Furthermore, we provide the quantitative results by considering three metrics including accuracy, intra-class distance and inter-class distance. In details, we set the number of layers from 2 to 32 with the step size as 2, and visualize the line chart of these comparison results in Figure 5 (g)∼(i). Figure 5 (g) shows the accuracy with regard to layers, which directly demonstrates the superiority of MotifNorm when GNNs go deeper. In order to characterize the disentangling capability of different normalizations, we calculate the intra-class distance and inter-class distance with layers increasing in Figure5 (h) and (i). As shown in these two figures, MotifNorm obtains lower intra-class distance and higher inter-class distance, which means that the proposed MotifNorm enjoys better disentangling ability. We provide more t-SNE visualizations of different layer number and different GNNs' backbones in Appendix B.3.

Table 3: Experimental results of different normalization methods on the graph prediction task. We use GCN, GAT and GIN as the basic backbones and set the number of layers as 4, 16 and 32. The best results on each dataset are highlighted with **boldface.**

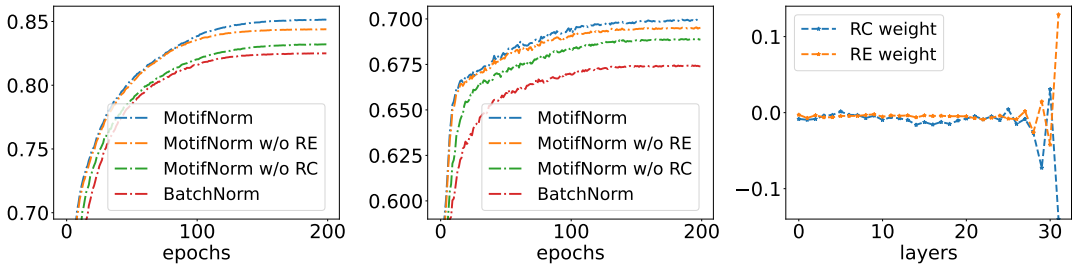
	Methods	ogbg-moltoxcast			ogbg-ppa			ogbg-molhiv			ZINC		
		4	16	32	4	16	32	4	16	32	4	16	32
GCN	NoNorm	61.13	59.34	56.08	66.14	66.54	66.17	76.01	71.90	60.59	0.643	0.690	0.748
	PairNorm	60.69	59.45	55.07	60.52	59.20	59.27	74.06	72.13	62.25	0.573	0.569	0.597
	NodeNorm	61.94	55.58	49.90	67.13	65.10	64.24	74.80	57.75	57.64	0.625	1.332	1.547
	MeanNorm	63.21	60.42	54.76	61.73	61.36	10.04	74.42	72.39	60.59	0.602	0.637	0.695
	GroupNorm	61.58	59.48	56.73	68.03	67.12	66.94	76.66	71.84	66.38	0.641	0.673	0.737
	GraphNorm	60.78	53.75	53.36	63.98	66.04	65.86	75.59	65.55	66.49	0.592	0.655	1.547
	BatchNorm	63.39	59.73	53.47	67.99	67.03	66.34	76.11	76.62	74.21	0.573	0.611	0.655
	ExpreNorm	64.97	57.91	57.82	68.11	66.86	66.44	76.05	76.75	72.36	0.564	0.570	0.646
	MotifNorm	**66.92**	**65.19**	**63.40**	**68.36**	**67.34**	**67.54**	**77.29**	**77.71**	**75.99**	**0.489**	**0.524**	**0.523**
GAT	NoNorm	62.61	50.84	50.12	65.63	09.48	09.48	76.71	57.38	50.64	0.714	1.541	1.547
	GraphNorm	60.53	52.79	53.22	62.54	64.63	65.28	75.30	73.86	64.03	0.576	1.254	1.537
	BatchNorm	63.31	53.39	53.24	67.35	67.08	66.46	76.07	76.87	73.74	0.585	0.624	0.643
	ExpreNorm	65.56	57.65	57.60	67.72	64.58	55.46	76.99	72.24	72.56	0.555	0.562	1.451
	MotifNorm	**66.57**	**64.04**	**58.26**	**68.59**	**67.95**	**67.02**	**77.36**	**77.08**	**76.70**	**0.495**	**0.517**	**0.522**
GIN	NoNorm	62.19	56.38	54.83	67.84	67.71	66.25	76.33	69.70	58.87	0.496	0.520	1.069
	GraphNorm	62.44	54.95	55.72	67.20	66.38	66.56	76.55	66.00	67.01	0.462	1.203	1.446
	BatchNorm	63.72	58.67	55.56	66.16	67.62	66.12	76.62	70.28	66.82	0.477	0.516	1.153
	ExpreNorm	65.98	57.80	56.56	67.13	67.94	67.15	76.23	69.97	70.96	0.438	0.482	1.157
	MotifNorm	**66.65**	**63.01**	**57.46**	**68.31**	**68.00**	**67.70**	**77.38**	**73.03**	**72.89**	**0.410**	**0.458**	**0.902**

Table 4: Experimental results of different normalization methods on the node prediction task and link prediction task. We use GCN, GAT and GIN as the basic backbones and set the number of layers as 4, 16 and 32. The best results on each dataset are highlighted with **boldface.**

	Settings	Pubmed			Citeseer			ogbn-proteins			ogbl-collab		
		4	16	32	4	16	32	4	16	32	4	16	32
GCN	NoNorm	76.16	54.67	45.58	64.89	39.19	27.37	69.16	63.24	63.15	35.38	22.11	15.24
	PairNorm	74.25	56.24	55.13	53.59	36.83	28.01	69.28	63.15	63.00	31.26	23.22	14.69
	NodeNorm	76.02	40.87	41.18	**66.18**	18.42	17.28	70.17	63.50	63.23	27.48	08.48	08.28
	MeanNorm	76.05	73.40	65.34	61.81	33.91	20.61	69.14	63.05	62.40	33.28	22.56	16.16
	GroupNorm	76.19	63.55	54.84	65.29	39.35	22.39	70.25	62.74	63.63	35.28	27.41	20.27
	BatchNorm	75.62	48.88	43.28	62.62	25.69	25.91	69.96	67.36	63.86	47.57	26.14	21.68
	MotifNorm	**77.08**	**76.66**	**67.81**	63.26	**58.52**	**41.69**	**71.69**	**68.66**	**68.05**	**51.65**	**50.01**	**47.65**
GraphSage	NoNorm	76.94	40.65	41.67	65.24	20.25	23.85	66.05	60.56	60.47	25.27	02.08	00.00
	PairNorm	72.78	53.02	45.90	64.24	30.19	24.12	62.29	60.53	60.32	41.72	16.88	12.44
	NodeNorm	77.22	40.64	40.64	**65.67**	18.76	18.05	64.48	62.63	61.89	19.74	02.57	02.62
	MeanNorm	76.68	58.70	47.48	63.33	24.21	21.65	63.69	61.03	52.06	46.17	21.54	13.16
	GroupNorm	76.83	40.42	43.49	64.80	21.84	22.91	68.09	61.58	60.60	45.43	23.98	15.43
	BatchNorm	75.49	45.11	42.74	63.45	27.49	25.89	63.75	62.96	61.54	47.05	23.01	14.89
	MotifNorm	**77.48**	**76.54**	**73.68**	63.90	**57.01**	**45.40**	**67.81**	**67.02**	**66.07**	**52.31**	**48.94**	**48.39**

4.3 More comparisons on the Other Eight Datasets

For graph classification task, we compare normalizations on ogbg-moltoxcast, ogbg-ppa, ogbg-molhiv and ZINC by using GCN, GAT and GIN with 4, 16, 32 layers. The ogbg-ppa and ogbg-moltoxcast are collected for multi-class and multi-task classification task. The ogbg-molhiv and ZINC are used for binary classification and graph regression task respectively. For node classification task, we conduct experiments on two social network datasets (Citeseer and Pubmed) and one protein–protein association network dataset (ogbn-proteins). For link prediction task, we use the ogbl-collab dataset, which the is collaboration network between authors indexed by MAG. By comprehensively considering the comparison results in Table 3 and Table 4, the mean values of 10 runs with different seeds (Appendix B.4 provides the standard deviations), we can conclude that MotifNorm generally improve the the graph expressivity of GNNs for graph prediction task and help alleviate the over-smoothing issue with the increase of layers.



(a) Training ROC-AUC (b) Valid ROC-AUC (c) Average weight in $w_{\mathrm{RC}}, w_{\mathrm{RE}}$

Figure 6: Abalation study of the Representation Calibration (RC) and Representation Enhancement (RE) operations in MotifNorm on the ogbg-moltoxcast dataset. Here we use GCN as the basic backbone to conduct the abalation study.

4.4 Ablation Study and Discussion

To explain the superior performance of MotifNorm, we perform extensive ablation studies to evaluate the contributions of its two key components, *i.e.*, representation calibration (RC) and representation enhancement (RE) operations. Firstly, we show in Figure 6 the ablation performance of GCN on ogbg-moltoxcast. Figure 6 (a) and (b) show the ROC-AUC results with regard to training epochs when the layer number is set to 4, which show that both RC and RE can improve the classification performance. Furthermore, by comparing these two figures, RC performs better than RE in terms of recognition results, which plugs at the beginning of BatchNorm with the graph instance-specific statistics embedded.

To further explore the significance of RC and RE at different layers, we compute the mean values of w_{RC} and w_{RE}, which are visualized in Figure 6 (c). As can be seen from the mean statistis of w_{RC}, w_{RE} among 32 layers' GCN, the absolute values of w_{RC} and w_{RE} become larger when the network goes deeper (especially in the last few layers), indicating that the structural information becomes more and more critical with the increase numbers of layers. Let's assume a deplorable condition of GNNs with very deep network layers, the representation power would then be thoroughly washed away due to the over-smoothing issue. In this case, MotifNorm can compensate for the loss of structural information to enhance the expressivity.

Discussion. The main contribution of this work is to propose a more expressive normalization module, which can be plugged into any GNN architecture. Unlike existing normalization methods that are usually task-specific and also without substructure information, the proposed method explicitly considers the subgraph information to strengthen the graph expressivity across various graph tasks. In particular, for the task of graph classification, MotifNorm extends GNNs beyond the 1-WL test in distinguishing k-regular graphs. On the other hand, when the number of GNNs' layers becomes larger, MotifNorm can help alleviate the over-smoothing problem and meanwhile maintain better discriminative power for the node-relevant predictions.

5 Conclusion

In this paper, we introduce a higher-expressivity normalization architecture with an abundance of graph structure-specific information embedded to generally improve GNNs' expressivities and representatives for various graph tasks. We first empirically disentangle the standard normalization into two stages, *i.e.*, centering & scaling (CS) and affine transformation (AT) operations, and then develop two skillful strategies to embed the subgraph structural information into CS and AT operations. Finally, we provide a theoretical analysis to support that MotifNorm can extend GNNs beyond the 1-WL test in distinguishing k-regular graphs and exemplify why it can help alleviate the over-smoothing issue when GNNs go deeper. Experimental results on 10 popular benchmarks show that our method is highly efficient and can generally improve the performance of GNNs for various graph tasks.

REFERENCES

Norah Alghamdi, Wennan Chang, Pengtao Dang, Xiaoyu Lu, Changlin Wan, Silpa Gampala, Zhi Huang, Jiashi Wang, Qin Ma, Yong Zang, et al. A graph neural network model to estimate cell-wise metabolic flux using single-cell rna-seq data. *Genome research*, 31(10):1867–1884, 2021.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Osman Boyaci, Mohammad Rasoul Narimani, Katherine R Davis, Muhammad Ismail, Thomas J Overbye, and Erchin Serpedin. Joint detection and localization of stealth false data injection attacks in smart grids using graph neural networks. *IEEE Transactions on Smart Grid*, 13(1): 807–819, 2021.

Khac-Hoai Nam Bui, Jiho Cho, and Hongsuk Yi. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence*, pp. 1–12, 2021.

Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, pp. 1204–1215. PMLR, 2021.

Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

Kaixuan Chen, Shunyu Liu, Na Yu, Rong Yan, Quan Zhang, Jie Song, Zunlei Feng, and Mingli Song. Distribution-aware graph representation learning for transient stability assessment of power system. *International Joint Conference on Neural Networks (IJCNN)*, 2022a.

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

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3837–3845, 2016.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2022a.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022b.

Shang-Hua Gao, Qi Han, Duo Li, Ming-Ming Cheng, and Pai Peng. Representative batch normalization with feature calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8669–8679, 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272, 2017.

Atika Gupta, Priya Matta, and Bhasker Pant. Graph neural network: Current state of art, challenges and applications. *Materials Today: Proceedings*, 46:10927–10932, 2021.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 2017.

Yunzhi Hao, Xinchao Wang, Xingen Wang, Xinyu Wang, Chun Chen, and Mingli Song. Walking with attention: Self-guided walking for heterogeneous graph embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Tackling over-smoothing for general graph convolutional networks. 2020.

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

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52 (7):1757–1768, 2012.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representation*, 2017.

Jure Leskovec. Fast neural subgraph matching and counting. *Stanford University*, CS224W: Machine Learning with Graphs, 2021. URL http://cs224w.stanford.edu/.

Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *International Conference on Computer Vision, ICCV*, pp. 9266–9275, 2019.

Jiachen Li, Siheng Chen, Xiaoyong Pan, Ye Yuan, and Hong-Bin Shen. Cell clustering for spatial transcriptomics data with graph neural networks. *Nature Computational Science*, 2(6):399–408, 2022.

Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4189–4196, 2021.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3538–3545, 2018.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 338–348, 2020.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations, ICLR 2020*, 2020.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

Mohammadamin Tavakoli, Aaron Mood, David Van Vranken, and Pierre Baldi. Quantum mechanics and machine learning synergies: graph attention neural networks to predict chemical reactivity. *Journal of Chemical Information and Modeling*, 62(9):2121–2132, 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representation*, 2018.

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

Asiri Wijesinghe and Qing Wang. A new perspective on" how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2022.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871, 2019.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5449–5458, 2018.

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

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

Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *Advances in Neural Information Processing Systems*, 33:20286–20296, 2020b.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations, ICLR*, 2020.

Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Min Wu, Tingjun Hou, and Mingli Song. Root-aligned smiles: a tight representation for chemical reaction prediction. *Chemical Science*, 13(31):9023–9034, 2022.

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

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

A THEOREM ANALYSIS

This section provides the corresponding proofs to support theorems in the main context.

A.1 PROOF FOR THEOREM 1

Theorem 1. *GNNs are as powerful the as 1-WL test in distinguishing non-isomorphic graphs while any two different subgraphs S_{v_i}, S_{v_j} are subtree-isomorphic (i.e., $S_{v_i} \simeq_{subtree} S_{v_j}$), or GNNs can map two different subgraphs into two different embeddings if and only if $S_{v_i} \not\simeq_{subtree} S_{v_j}$.*

Proof. We provide theoretical analysis by comparing the difference between the formulation of the message-passing GNNs and the 1-WL test, and detail as follow:

For graph representation learning using GNNs, node features in a graph are usually learned by following the neighborhood aggregation and representation updating schemes. In general, the formulation of the message-passing GNNs' convolution can be represented as:

$$\texttt{GNNs} : h_v^{(t)} = \mathcal{M}(h_v^{t-1}, \mathcal{A}\{h_u^{t-1} | u \in \mathcal{N}(v)\}), \tag{8}$$

where $\mathcal{A}$ is the aggregation function, and $\mathcal{M}$ is a representation updating function. For a clear comparison with 1-WL test, we copy the formulation of 1-WL from Eq.(1) and add notation for distinguishing:

$$\texttt{1-WL} : h_v^{(t)} = \texttt{Hash}(h_v^{t-1}, \mathcal{A}\{h_u^{t-1} | u \in \mathcal{N}(v)\}). \tag{9}$$

Let us compare the formulation of above Eq.(8) and Eq.(9), and we can find that the difference between the two equations is that the hash function $\texttt{Hash}(\cdot)$ and updating function $\mathcal{M}(\cdot)$. Here, the updating function $\mathcal{M}$ in GNNs is not always injective, so that may transform two different samples into the same representation, which is the main reason that GNNs are at most as powerful as the 1-WL test for graph isomorphic issues. To prove this theorem, we take the subtree-isomorphic issue (*e.g.*, two node-induced subgraphs, S_{v_i}, S_{v_i}) for example and exemplify as follow:

- If $S_{v_i} \simeq_{\texttt{subtree}} S_{v_j}$, the simple neighborhood aggregation operation just concerns 1-hop information, resulting in both GNNs and 1-WL test can not distinguish these two substructures. In this case, GNNs are as powerful as 1-WL test in distinguishing non-isomorphic graphs.
- If $S_{v_i} \not\simeq_{\texttt{subtree}} S_{v_j}$, the neighborhood aggregation operation can obtain two different multisets. However, $\mathcal{M}$ may transform two different multisets into the same representation, which means that GNNs are **not** as powerful as 1-WL test in distinguishing non-isomorphic graphs.
- If $S_{v_i} \not\simeq_{\texttt{subtree}} S_{v_j}$ and $\mathcal{M}$ map two different substructures into different representations (*i.e.*, $f(S_{v_i}) \neq f(S_{v_j})$), which means that $\mathcal{M}$ distinguish these two substructures like Hash in 1-WL. In this case, GNNs are as powerful as 1-WL test in distinguishing non-isomorphic graphs.

To this end, we analyzed the existing conditions of GNNs as powerful as 1-WL. The first and third items are the statement of Theorem 1. Wijesinghe & Wang (2022) provided a similar theorem from the perspective of the WL iteration over multisets, interested readers please refer to that for details.

The proof is complete. □

A.2 PROOF FOR THEOREM 2

Theorem 2. *RC operation calibrates input features for the following CS, considering the motif-induced structural weights and the graph instance-specific statistics, which benefits strengthening the structural power in node representations, along with stabling and accelerating GNNs' training.*

Proof. Embedding the motif-induced weights into normalization, along with the mean statistics of each graph, strengthens the structural power in node representations and balances the distribution differences among the batch graphs and graph instance-specific statistics. Especially, the running means count the mean statistics over the training set, but it is not always suitable for valid and

test sets. That is to say, directly centering features with the statistics on the training set may cause the misplacement of informative representations. The second term proposed in Eq.(5) makes normalization operation less reliant on the running means and balances the distribution differences by considering the graph instance-specific statistics. The details are as follows:

To clearly understand the effect of $\mathrm{H}_{\mathtt{SA}}$, we preferably remove the $\mathrm{M}_{\mathtt{RC}}$ term and assume the number of nodes in each graph is consistent. The formulation of Eq.(5) can be rewritten as:

$$\mathrm{H}_{\mathtt{RC}} = \mathrm{H} + w_{\mathtt{RC}} \odot \mathrm{H}_{\mathtt{SA}}, \tag{10}$$

where $w_{\mathtt{RC}} \odot \mathrm{H}_{\mathtt{SA}}$ introduces the current graph's instance-specific information, *i.e.*, mean representations in each graph. $w_{\mathtt{RC}}$ is a learnable weight balancing mini-batch and instance-specific statistics. The expectation of input features after $\mathtt{RC}$, *i.e.*, $\mathbb{E}(\mathrm{H}_{\mathtt{RC}})$, can be represented as:

$$\mathbb{E}(\mathrm{H}_{\mathtt{RC}}) = (1 + w_{\mathtt{RC}}) \odot \mathbb{E}(\mathrm{H}). \tag{11}$$

Let us respectively consider the following centering operation of normalization for the original input H and the feature matrix $\mathrm{H}_{\mathtt{RC}}$ after $\mathtt{RC}$ operation,

$$\begin{aligned} \mathrm{H}_{\mathtt{Center\text{-}In}} &= \mathrm{H} - \mathbb{E}(\mathrm{H}), \\ \mathrm{H}_{\mathtt{Center\text{-}RC}} &= \mathrm{H}_{\mathtt{RC}} - \mathbb{E}(\mathrm{H}_{\mathtt{RC}}), \end{aligned} \tag{12}$$

where $\mathrm{H}_{\mathtt{Center\text{-}In}}$ and $\mathrm{H}_{\mathtt{Center\text{-}RC}}$ denote the centering operation on H and $\mathrm{H}_{\mathtt{RC}}$. To compare the difference between these two centralized features, we perform:

$$\begin{aligned} \mathrm{H}_{\mathtt{Center\text{-}RC}} - \mathrm{H}_{\mathtt{Center\text{-}In}} &= (\mathrm{H}_{\mathtt{RC}} - \mathbb{E}(\mathrm{H}_{\mathtt{RC}})) - (\mathrm{H} - \mathbb{E}(\mathrm{H})) \\ &= \mathrm{H} + w_{\mathtt{RC}} \odot \mathrm{H}_{\mathtt{SA}} - (1 + w_{\mathtt{RC}}) \odot \mathbb{E}(\mathrm{H}) - (\mathrm{H} - \mathbb{E}(\mathrm{H})) \\ &= w_{\mathtt{RC}} \odot (\mathrm{H}_{\mathtt{SA}} - \mathbb{E}(\mathrm{H})). \end{aligned} \tag{13}$$

While values in $w_{\mathtt{RC}}$ are close to zero, the centering operation still relies on running statistics over the training set. On the other hand, the importance of graph instance-specific statistics grows when the absolute value of $w_{\mathtt{RC}}$ becomes larger. Here, we ignore the affine transformation operation and assume values larger than the running mean, kept after the following activation layer, are important information for representations, and vice versa. In case of $w_{\mathtt{RC}} > 0$, while $\mathrm{H}_{\mathtt{SA}} > \mathbb{E}(\mathrm{H})$, more important information tends to be preserved, and vice versa. In case of $w_{\mathtt{RC}} < 0$, while $\mathrm{H}_{\mathtt{SA}} > \mathbb{E}(\mathrm{H})$, the noisy features tend to be weakened, and vice versa. A similar analysis in `BatchNorm2D` has been provided in (Gao et al., 2021), interested readers please refer to that for details. Based on the above analysis, Eq.(5) with motif information embedded will not hurt this advantage because all motif-induced weights are positive numbers.

Furthermore, Eq.(10) is a particular type of graph decentering operation, which has been proven to accelerate `GNNs`' training, and please kindly refer to (Cai et al., 2021) for details.

Finally, we talk about the design of representation calibration factor $\mathrm{M}_{\mathtt{RC}} = \mathrm{M}_{\mathtt{SN}} \cdot \mathrm{M}_{\mathcal{G}}$, which is a normalization for motif-induced weights $\mathrm{M}_{\mathcal{G}}$. If we directly adopt the original weights, existing many unequal large values, it will make training oscillating. Thus, the normalization is essential for $\mathrm{M}_{\mathcal{G}}$. However, if we just perform summation-normalization in an arbitrary graph (*i.e.*, $\mathrm{M}_{\mathtt{SN}}$), it will not distinguish two graphs with the same nodes but different degrees, *e.g.*, four graphs in Figure 7 where each weight will become 1/8. To this end, we design the above normalization technique to strengthen the motif power for the representation calibration operation.

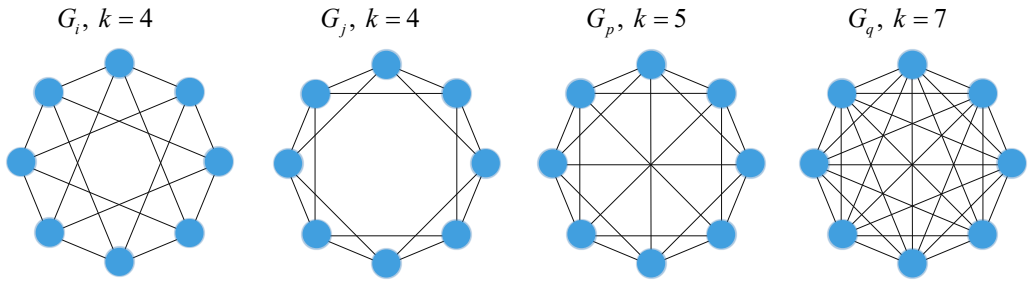


Figure 7: The illustration of four k-regular graphs with the same nodes but different structures. When directly performing the summation-normalization operation on motif-induced information, all weights will be equal to 1/8.

The proof is complete. □

A.3 PROOF FOR THEOREM 3

Theorem 3. *`MotifNorm` extends `GNNs` beyond the `1-WL` test in distinguishing k-regular graphs and helps alleviate the over-smoothing issue when `GNNs`' layers become deeper.*

Proof. Since `GNNs` are as powerful as `1-WL` when two graphs are subtree-isomorphic (*i.e.*, the first item in the proof of Theorem 1). An arbitrary `GNNs` equipped with `MotifNorm` are more expressive than `1-WL` in distinguishing k-regular graphs because it can at least distinguish two subtree-isomorphic structures S_{v_1},S_{v_2} in Figure 2. Furthermore, while two different samples become similar, `MotifNorm` will make the node representations scattered and decentralized by embedding substructural weights, and thus help alleviate the over-smoothing issue.

More detailed analysis are as follow:

For the non-isomorphic graphs distinguishing issue, we follow the analysis procedure in the proof of Theorem 1, *i.e.*, taking subtree-isomorphic substructures, for example.

- When $S_{v_i} \simeq_{\texttt{subtree}} S_{v_j}$, `GNNs`' expressivities are as powerful as `1-WL` test in distinguishing non-isomorphic graphs. However, `GNNs` with the motif-induced information embedded, *i.e.*, performing `MotifNorm`, can at least distinguish two substructures in Figure 2, where `1-WL` misses the distinguishable power. In this case, `GNNs` are more expressive than `1-WL` test.
- When $S_{v_i} \not\simeq_{\texttt{subtree}} S_{v_j}$ and $\mathcal{M}$ transforms two different substructures into the same representation, `GNNs` are **not** as powerful as `1-WL` test in distinguishing non-isomorphic graphs. However, `MotfiNorm` can improve the graph expressivity of `GNNs` as powerful as `1-WL` except for some exceptional cases, *e.g.*, motif-induced weights of S_{v_i}, S_{v_j} are consistent.
- When $S_{v_i} \not\simeq_{\texttt{subtree}} S_{v_j}$ and $\mathcal{M}$ maps two different substructures into different representations (*i.e.*, $f(S_{v_i}) \neq f(S_{v_j})$), `GNNs`' expressivities are as powerful as `1-WL` test in distinguishing non-isomorphic graphs. `GNNs` maintain this advantage with `MotifNorm` adopted except that $f(S_{v_i})$, $f(S_{v_j})$ are linearly dependent, *i.e.*, $f(S_{v_i}) = kf(S_{v_j})$, and $\xi(S_{v_i}) = \frac{1}{k}\xi(S_{v_j})$ at the same time.

The first item is the proof of the Theorem 3, *i.e.*, `MotifNorm` can extend `GNNs` beyond the `1-WL` test in distinguishing k-regular graphs, and other items analyze graph expressivities of `MotifNorm` for `GNNs` in distinguishing non-isomorphic graphs.

For the over-smoothing issue in `GNNs` with layers increasing, `MotifNorm` captures structural connectivity between a target vertex and its neighbors. Thus, a neighbor whose structural connectivity is weak would be calibrated and enhanced with little information to the target vertex, and vice versa. Such operation tends to maintain the structural information in the resulting representations, along with `GNNs` going deeper. Even two different node representations become similar, `MotifNorm` can scatter them with different weights, which helps alleviate the over-smoothing issue.

Suppose that node u and node v obtain extremely similar embeddings after L-layer `GNNs` that $\|\mathbf{h}_u - \mathbf{h}_v\|_2 \leq \epsilon$, where $|\mathbf{h}_u\|_2 = \|\mathbf{h}_v\|_2 = 1$ for simplicity. And the motif-induced information scores between u and v differs a considerable margin that $\|\xi(u) - \xi(v)\|_2 \geq \delta + \epsilon$, where $\xi(\cdot) \leq 1$ for simplicity. By mutiplying the similar node embeddings with the different motif-induced information scores, we get that

$$\begin{aligned}
\|\xi(u)\mathbf{h}_u - \xi(v)\mathbf{h}_v\|_2 &= \|\xi(u)\mathbf{h}_u - \xi(u)\mathbf{h}_v + \xi(u)\mathbf{h}_v - \xi(v)\mathbf{h}_u\|_2 \\
&= \|\underbrace{\xi(u)(\mathbf{h}_u - \mathbf{h}_v)}_{\approx 0} + (\xi(u) - \xi(v))\mathbf{h}_v\|_2 \\
&\geq -\|\xi(u)(\mathbf{h}_u - \mathbf{h}_v)\|_2 + \|(\xi(u) - \xi(v)\mathbf{h}_v\|_2 \\
&\geq -\epsilon + \delta + \epsilon = \delta.
\end{aligned} \tag{14}$$

It states that our motif-induced information scores could differentiate two nodes when their node embeddings become extremely similar after L-layer `GNNs`.

The proof of Theorem 3 is complete. □

B EXPERIMENTAL DETAILS

B.1 MORE DETAILS OF BENCHMARK DATASETS AND BASELINE METHODS

Benchmark Datasets. For the task of graph property prediction, we select `IMDB-BINARY`, `ogbg-moltoxcast`, `ogbg-ppa`, `ogbg-molhiv` and `ZINC` datasets. The `IMDB-BINARY` is a k-regular dataset for binary classification task, which means that each node has a degree k. The `ogbg-moltoxcast` and `ogbg-ppa` are collected for multi-task and multi-class classification task, where the number of the tasks and classes are 617 and 37 respectively. The `ogbg-molhiv` is a molecule dataset that used for binary classification task, but the ouput dimension of the end-to-end GNNs is 1 because its metric is ROC-AUC. The `ZINC` is the real-world molecule dataset for the example reconstruction. In this paper, we follow the work in (Dwivedi et al., 2022a) to use `ZINC` for the task of graph regression. These graph prediction datasets are from (Morris et al., 2020; Hu et al., 2020; Irwin et al., 2012) respectively. For the node level prediction, we select four benchmark datasets including `Cora`, `Citeseer`, `Pubmed` and `ogbn-proteins`. The first three datasets are the social network and the last one is a protein–protein association network dataset. For the evalutation of link property prediction, we select `ogbl-collab` dataset in this paper. These node and link prediction datasets are from (Kipf & Welling, 2017; Hu et al., 2020) respectively. More detailed dataset information is provided in 5.

Baseline Methods. To evalute our proposed `MotifNorm` module, we need to compare other normalization methods adopted in `GNNs` for various graph tasks, including `BatchNorm` (Ioffe & Szegedy, 2015), `GraphNorm` (Cai et al., 2021), `ExpreNorm` (Dwivedi et al., 2022a) for graph property prediction, and `PairNorm` (Zhao & Akoglu, 2020), `NodeNorm` (Zhou et al., 2021), `MeanNorm` (Yang et al., 2020a), `GroupNorm` (Zhou et al., 2020) for node and link property prediction. A part of these normalization methods are provided in (Chen et al., 2022b), and other source codes are provided by authors. For the backbone `GNNs`, we consider the most popuare message-passing architectures such as `GCN` (Kipf & Welling, 2017), `GAT` (Veličković et al., 2018), `GIN` (Xu et al., 2019), `SGC` (Wu et al., 2019) and `GraphSage` (Hamilton et al., 2017). Specially, we will compare our `MotifNorm` with all above normlization modules. For the network architectures, we follow `CNA` architecture, *i.e.*, convolution, normalization and activation. In this paper, we do not adopt any skills like dropedge (Huang et al., 2020; Rong et al., 2020), residual connection (Xu et al., 2018; Li et al., 2019; Liu et al., 2020), etc.

Experiment Setting. For different datasets, we provide more detailed statistics information in Table 5. The embedding dimension in each hidden layer on all datasets is set as 128. We optim the `GNNs`' architectures using `torch.optim.lr_scheduler.ReduceLROnPlateau` by setting patience step as 10 or 15 to reduce learning rate. The learning rate is $1e-3$ for graph classification, and $1e-2$ for node, link predictions. When the learning rate reduces to $1e-5$, the training will be terminated. More detailed statistics of experimental settings are provided in Table 6. Specially, we split the `IMDB-BINARY` dataset into train-vallid-test format using a hierarchical architecture. In details, we first segment this dataset according to the edge density information into 10 set (i.e., the edge density $\in \{0.0-0.1\}, \cup, \{0.1-0.2\}..., \{0.9-1.0\}$, and then sort the graphs using the average degree information. Finally, we select the samples in each segment using a fix step size as valid and test samples. The statistic information for splitting the valid and test set of `Label-0` and `Label-1` on `IMDB-BINARY` is provide in Table 7. By adopting this splitting scheme, distribution differences among train, valid and test sets are weakened (Experiment results show this contribution fact but without a theoretical basis now). The splitting details are implemented in the `datasets/dgl_imdb_dataset.py`. To reproduce the comparison results using a single layer of `MLP` and `GIN`, the dropout is set to 0.0 and warming up the learning rate from 0.0 to $1e-3$ at the first 50 epoches. When layer is equal to 4, the doupout at the input layer is selected in $\{0.3, 0.4, 0.5\}$ (*i.e.*, -init_dp in the code), and hidden layer is set to 0.5. To draw the Figure 3, we remove the warmup operation for learning rate (*i.e.*, remove –lr_warmup in the shell files). For the `ogbg-ppa` dataset, we need consider edge information for `GNNs`' convolution (*i.e.*, -econv in the code).

Implementation. `MotifNorm` needs to process the motif-induced weight into datasets and then load this processing information to embed structural information into node representations. Especially for the node-relevant classification, node representations need to contain the same power before `MotifNorm`. Thus, we perform the l-2 normalization to ensure their power are consistent. The two scripts are at: `datasets/preprocess.py` and `modules/norm/motifnorm.py`.

Table 5: The statistic information of 10 benchmark datasets.

Dataset Name	Dataset Type	Task Type	#Graphs	Avg. #Nodes	Avg. #Edges
IMDB-BINARY	molecular	Graph classification	1,000	19.8	193.1
ogbg-toxcast	molecular	Graph classification	8,576	18.8	19.3
ogbg-ppa	protein association	Graph classification	158,100	243.4	2,266.1
ogbg-molhiv	molecular	Graph classification	41,127	25.5	27.5
ZINC	molecular	Graph regression	10,000	23.2	49.8
Cora	social	Node classification	1	2,708	5,429
Citeseer	social	Node classification	1	3,327	4,732
Pubmed	social	Node classification	1	19,717	44,338
ogbn-proteins	proteins	Node classification	1	132,534	39,561,252
ogbl-collab	social	Link classification	1	235,868	1,285,465

Table 6: The detailed experimental settings of GNNs on various graph-structured tasks.

Name	Metrics	Edge Conv.	Layers	LR	Batch Size	InitDim.	HiDim.	Weight decay	Dropout
IMDB-BINARY	ROC-AUC	False	$1, 4$	$1e-3$	32	128	128	0.0	0.0, 0.5
ogbg-toxcast	ROC-AUC	False	$4, 16, 32$	$1e-3$	128	9	128	0.0	0.5
ogbg-ppa	Accuracy	True	$4, 16, 32$	$1e-3$	256,128	7	128	0.0	0.5
ogbg-molhiv	ROC-AUC	False	$4, 16, 32$	$1e-3$	256	9	128	0.0	0.5
ZINC	MAE	False	$4, 16, 32$	$1e-3$	128	1	128	0.0	0.5
Cora	Accuracy	False	$[0; 2; 32]$	$1e-2$	$--$	1433	128	0.0	0.5
Citeseer	Accuracy	False	$4, 16, 32$	$1e-2$	$--$	3703	128	0.0	0.5
Pubmed	Accuracy	False	$4, 16, 32$	$1e-2$	$--$	500	128	0.0	0.5
ogbn-proteins	ROC-AUC	False	$4, 16, 32$	$1e-2$	$--$	8	128	0.0	0.5
ogbl-collab	Hits@50	False	$4, 16, 32$	$1e-2$	64×1024	128	128	0.0	0.0

Table 7: The statistic information for splitting IMDB-BINARY.

	0.0−0.1	0.1−0.2	0.2−0.3	0.3−0.4	0.4−0.5	0.5−0.6	0.6−0.7	0.7−0.8	0.8−0.9	0.9−1.0	1.0	Total Num.
Label-0	1	17	60	89	49	117	52	12	15	7	81	500
Label-1	0	22	81	145	58	97	30	8	1	0	58	500
Total Label	1	39	141	234	107	214	82	20	16	7	139	1000
Steps	-	9	8	7	6	5	4	3	2	2	6	
Label-0 Sel.	0	2	14	28	16	46	30	8	14	6	26	190
Label-1 Sel.	0	4	20	40	18	38	14	4	0	0	18	156
Total Sel.	0	6	34	68	34	84	44	12	14	6	44	346

B.2 More Comparisons on Graph Isomorphism Test

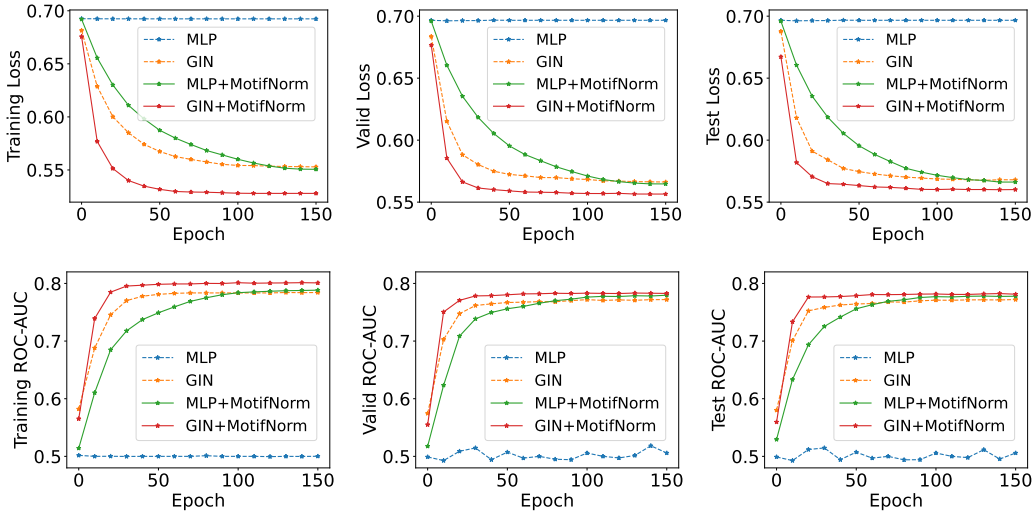


Figure 8: Learning curves of one-layer `MLP`, `GIN`, `MLP` + `MotifNorm` and `GIN` + `MotifNorm` on `IMDB-BINARY` dataset with various k-regular graphs.

Table 8: Experimental results on `IMDB-BINARY` dataset with various k-regular graphs. The best results under different backbones are highlighted with **boldface.**

	Normalization			L	ROC-AUC			LOSS			#Epochs
	NoNorm	BatchNorm	MotifNorm		Test Split	Valid Split	Train Split	Test Split	Valid Split	Train Split	
`MLP`	✓			1	56.66 ± 1.09	57.05 ± 0.81	50.40 ± 0.04	0.6961	0.6961	0.6927	36.40
		✓		1	56.86 ± 0.72	57.83 ± 0.84	51.38 ± 2.69	0.6954	0.6954	0.6923	112.90
			✓	1	$\mathbf{78.16 \pm 0.52}$	$\mathbf{78.69 \pm 0.44}$	$\mathbf{79.05 \pm 0.34}$	0.5631	0.5581	0.5519	190.70
`GIN`	✓			1	77.40 ± 0.20	77.69 ± 0.13	78.95 ± 0.16	0.5684	0.5658	0.5525	196.60
		✓		1	77.71 ± 0.19	78.11 ± 0.10	79.65 ± 0.23	0.5655	0.5593	0.5391	152.80
			✓	1	$\mathbf{78.42 \pm 0.15}$	$\mathbf{78.89 \pm 0.15}$	$\mathbf{80.24 \pm 0.18}$	0.5584	0.5565	0.5281	178.20
`GSN`		✓		1	77.50 ± 0.18	77.54 ± 0.16	78.67 ± 0.17	0.5696	0.5688	0.5570	67.20
`GraphSNN`		✓		1	77.52 ± 0.16	77.61 ± 0.18	78.90 ± 0.15	0.5691	0.5671	0.5520	89.50
`GIN`	✓			4	78.22 ± 0.41	78.27 ± 0.45	80.03 ± 0.30	0.5602	0.5553	0.5304	154.60
		✓		4	78.41 ± 0.21	78.76 ± 0.17	80.10 ± 0.14	0.5625	0.5590	0.5325	161.00
			✓	4	$\mathbf{79.55 \pm 0.35}$	$\mathbf{79.62 \pm 0.38}$	$\mathbf{81.50 \pm 0.48}$	0.5539	0.5510	0.5133	133.20
`GCN`	✓			4	68.99 ± 1.38	69.08 ± 1.81	68.09 ± 1.81	0.6920	0.6921	0.6850	33.10
		✓		4	76.75 ± 1.31	76.96 ± 0.69	75.70 ± 1.05	0.5755	0.5640	0.5838	153.80
			✓	4	$\mathbf{78.78 \pm 1.01}$	$\mathbf{79.03 \pm 0.61}$	$\mathbf{78.69 \pm 0.84}$	0.5597	0.5504	0.5493	61.90
`GAT`	✓			4	69.07 ± 1.59	69.78 ± 1.65	66.78 ± 1.68	0.6904	0.6891	0.6887	33.10
		✓		4	75.10 ± 1.51	75.95 ± 1.27	75.26 ± 0.87	0.5866	0.5852	0.5978	156.00
			✓	4	$\mathbf{78.87 \pm 0.80}$	$\mathbf{79.20 \pm 0.56}$	$\mathbf{79.19 \pm 0.78}$	0.5559	0.5448	0.5504	122.00
`GraphSage`	✓			4	64.35 ± 4.36	65.97 ± 4.85	63.49 ± 2.93	0.6923	0.6921	0.6918	21.00
		✓		4	75.06 ± 1.77	76.12 ± 1.31	75.95 ± 1.05	0.5819	0.5748	0.5805	146.00
			✓	4	$\mathbf{78.93 \pm 0.99}$	$\mathbf{79.02 \pm 0.64}$	$\mathbf{79.23 \pm 0.49}$	0.5534	0.5481	0.5447	74.00
`SGC`	✓			4	62.83 ± 2.66	66.19 ± 2.86	62.30 ± 2.37	0.6951	0.6941	0.6876	142.00
		✓		4	70.94 ± 1.33	73.51 ± 1.02	71.20 ± 0.65	0.6231	0.6044	0.6109	112.00
			✓	4	$\mathbf{78.85 \pm 0.70}$	$\mathbf{78.53 \pm 0.83}$	$\mathbf{78.61 \pm 0.63}$	0.5577	0.5556	0.5466	90.00
`GSN`		✓		4	78.90 ± 0.63	79.28 ± 0.70	80.70 ± 0.37	0.5555	0.5543	0.5377	94.20
`GraphSNN`		✓		4	79.16 ± 0.67	79.35 ± 0.82	80.74 ± 0.44	0.5541	0.5530	0.5356	73.60

B.3 More Comparisons on the Over-smoothing Issue

This section provide the illustration of GCN (Kipf & Welling, 2017) and GraphSage (Hamilton et al., 2017) with different six existent normalizations on Cora dataset.

Given the node embedding $\mathrm{X} = [x_1, x_2, ..., x_n]^{\mathrm{T}} \in \mathbb{R}^{n \times d}$, where $x_i \in \mathbb{R}^d$ denotes the node embedding at layer i, these normalization methods includiong PairNorm (Zhao & Akoglu, 2020), NodeNorm (Zhou et al., 2021), MeanNorm (Yang et al., 2020a), GroupNorm (Zhou et al., 2020) and BatchNorm (Ioffe & Szegedy, 2015) are depicted as:

PairNorm (Zhao & Akoglu, 2020).

$$\begin{aligned} \tilde{x}_i &= x_i - \frac{1}{n}\sum_{i=1}^{n} x_i, \\ \texttt{PairNorm}(x_i, s) &= \frac{s \cdot \tilde{x}_i}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} ||\tilde{x}_i||_2^2}}. \end{aligned} \tag{15}$$

NodeNorm (Zhou et al., 2021).

$$\texttt{NodeNorm}(x_i, p) = \frac{x_i}{\mathrm{std}(x_i)^{\frac{1}{p}}}. \tag{16}$$

MeanNorm (Yang et al., 2020a).

$$\texttt{MeanNorm}(x_{(k)}) = x_{(k)} - \mathbb{E}[x_{(k)}]. \tag{17}$$

BatchNorm (Ioffe & Szegedy, 2015).

$$\texttt{BatchNorm}(\mathrm{X}) = \frac{\mathrm{X} - \mathbb{E}(\mathrm{X})}{\sqrt{\mathbb{D}(\mathrm{X}) + \epsilon}} \odot \gamma + \beta. \tag{18}$$

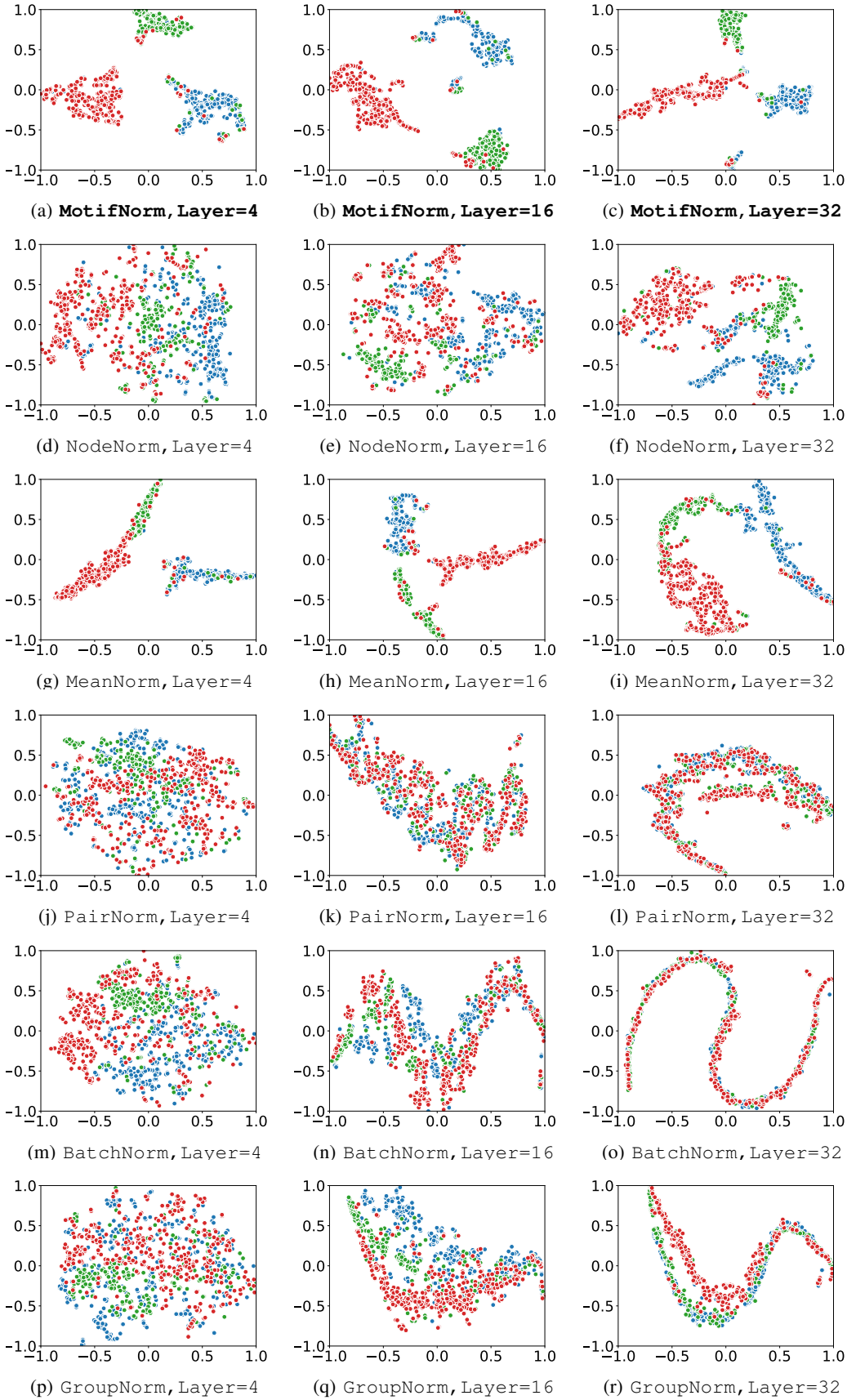
GroupNorm (Zhou et al., 2020).

$$\begin{aligned} \texttt{GroupNorm}(\mathrm{X}; G, \lambda) &= \mathrm{X} + \lambda \cdot \sum_{g=1}^{G} \texttt{BatchNorm}(\tilde{\mathrm{X}}_g), \\ where\ \ \tilde{\mathrm{X}}_g &= \mathrm{softmax}(\mathrm{X} \cdot \mathrm{U})[:, g] \circ \mathrm{X}. \end{aligned} \tag{19}$$

MotifNorm please refer to 7 for details.

Here $x_{(i)} \in \mathbb{R}^n$ denotes the i-th column of X, s in PairNorm is a hyperparameter controlling the average pair-wise variance and we choose $s = 1$ in our case. p in NodeNorm denotes the normalization order and our paper uses $p = 2$. G means the number of groups, and λ is the skip connection coefficient used in GroupNorm. The G is selected in $\{4, 6, 8\}$ and skip weight is $1e-2$ in this paper. More notations please refer to (Chen et al., 2022b) for details.

The t-SNE illustrations of different normalizations embedded in GCN and GraphSage are provided in Figure 9 and 10 respectively, where the number of layers are 4, 16, 32. Figure 11 shows the over-smoothing phenomenon with the layers increasing using GraphSage, where the number of layers is from 16 to 32 with the step size as 2.



(a) **MotifNorm, Layer=4** (b) **MotifNorm, Layer=16** (c) **MotifNorm, Layer=32**

(d) NodeNorm, Layer=4 (e) NodeNorm, Layer=16 (f) NodeNorm, Layer=32

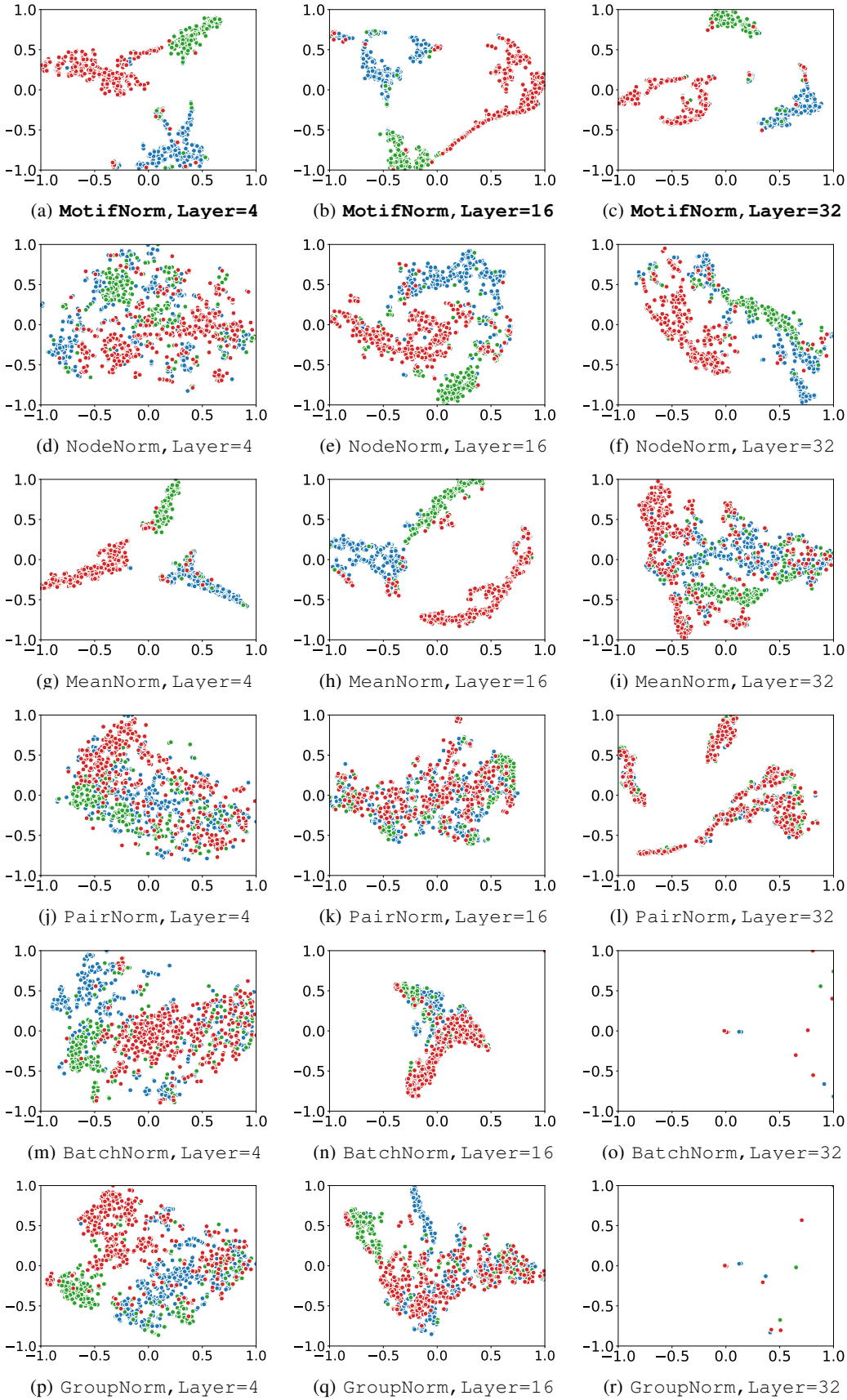
(g) MeanNorm, Layer=4 (h) MeanNorm, Layer=16 (i) MeanNorm, Layer=32

(j) PairNorm, Layer=4 (k) PairNorm, Layer=16 (l) PairNorm, Layer=32

(m) BatchNorm, Layer=4 (n) BatchNorm, Layer=16 (o) BatchNorm, Layer=32

(p) GroupNorm, Layer=4 (q) GroupNorm, Layer=16 (r) GroupNorm, Layer=32

Figure 9: The t-SNE visualization using GCN with different normalization methods on Cora dataset.



(a) **MotifNorm, Layer=4** (b) **MotifNorm, Layer=16** (c) **MotifNorm, Layer=32**

(d) NodeNorm, Layer=4 (e) NodeNorm, Layer=16 (f) NodeNorm, Layer=32

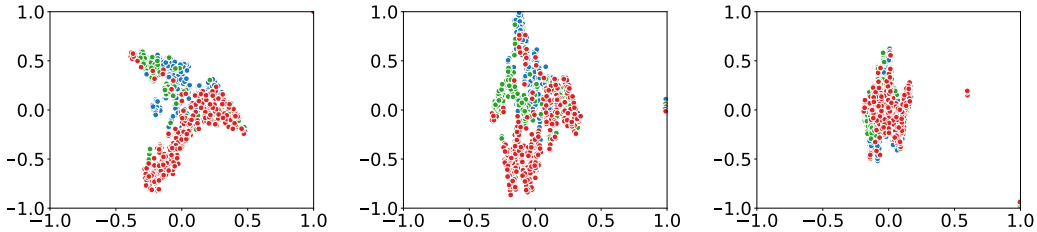
(g) MeanNorm, Layer=4 (h) MeanNorm, Layer=16 (i) MeanNorm, Layer=32

(j) PairNorm, Layer=4 (k) PairNorm, Layer=16 (l) PairNorm, Layer=32

(m) BatchNorm, Layer=4 (n) BatchNorm, Layer=16 (o) BatchNorm, Layer=32

(p) GroupNorm, Layer=4 (q) GroupNorm, Layer=16 (r) GroupNorm, Layer=32

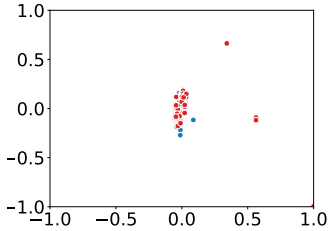
Figure 10: The t-SNE visualization using GraphSage with different normalization methods on Cora dataset.



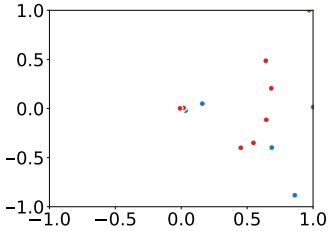
(a) BatchNorm, Layer=16

(b) BatchNorm, Layer=18

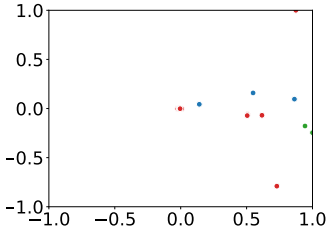
(c) BatchNorm, Layer=20



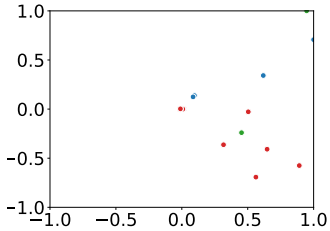
(d) BatchNorm, Layer=22



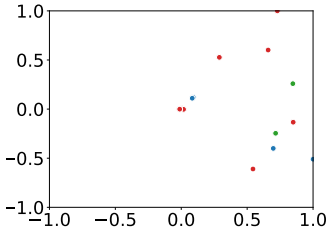
(e) BatchNorm, Layer=24



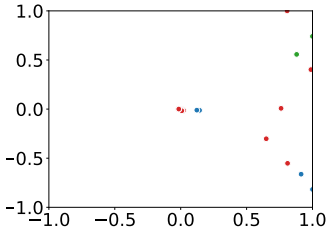
(f) BatchNorm, Layer=26



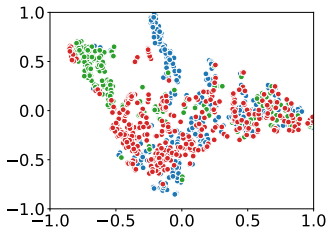
(g) BatchNorm, Layer=28



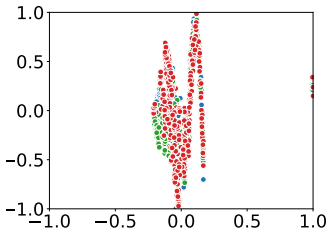
(h) BatchNorm, Layer=30



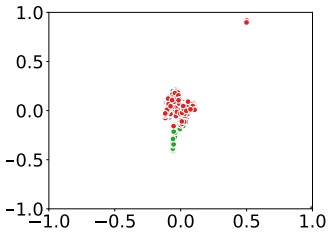
(i) BatchNorm, Layer=32



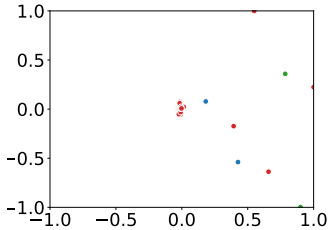
(j) GroupNorm, Layer=16



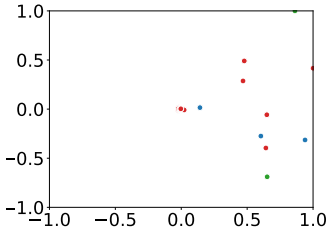
(k) GroupNorm, Layer=18



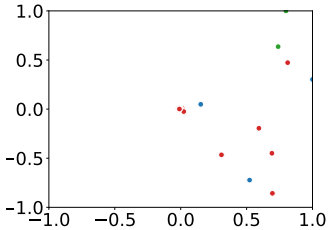
(l) GroupNorm, Layer=20



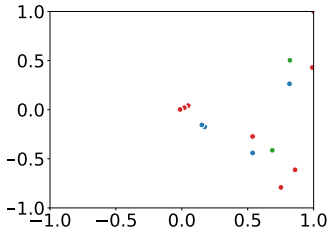
(m) GroupNorm, Layer=22



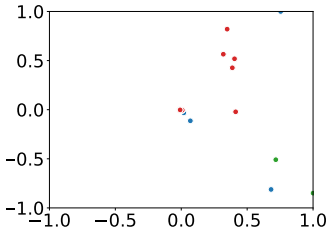
(n) GroupNorm, Layer=24



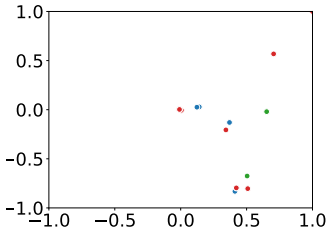
(o) GroupNorm, Layer=26



(p) GroupNorm, Layer=28



(q) GroupNorm, Layer=30



(r) GroupNorm, Layer=32

Figure 11: The t-SNE visualization using GraphSage with BatchNorm and GroupNorm on Cora dataset.

B.4 More Statistics on the Other Eight Datasets

Table 9: The **standard deviations** of the performance (Table 3) on the graph prediction task. We use GCN, GAT and GIN as the basic backbones and set the number of layers as 4, 16 and 32.

	Settings	ogbg-toxcast			ogbg-ppa			ogbg-molhiv			ZINC		
		4	16	32	4	16	32	4	16	32	4	16	32
GCN	NoNorm	0.468	0.782	2.364	0.951	0.905	0.444	0.924	0.926	2.546	0.013	0.014	0.015
	PairNorm	0.865	0.480	1.340	0.672	1.017	0.176	0.669	1.931	5.213	0.004	0.005	0.006
	NodeNorm	0.555	2.437	0.636	0.730	0.662	1.240	1.152	1.244	1.033	0.005	0.274	0.001
	MeanNorm	0.700	0.850	0.734	0.758	1.221	1.008	0.696	1.204	2.546	0.011	0.015	0.011
	GroupNorm	0.579	0.733	2.252	0.585	0.374	0.172	1.081	1.466	2.416	0.012	0.010	0.016
	GraphNorm	1.034	0.896	1.089	0.848	0.524	0.709	0.993	4.151	1.642	0.009	0.029	0.001
	BatchNorm	1.036	2.733	1.363	0.594	0.975	0.567	0.981	1.793	2.284	0.016	0.017	0.025
	ExpreNorm	0.416	0.551	0.298	0.482	0.813	0.909	0.947	1.382	0.497	0.009	0.015	0.036
	MotifNorm	0.698	0.482	0.959	0.189	0.505	0.166	0.984	1.298	2.165	0.010	0.012	0.011
GAT	NoNorm	0.449	1.400	0.373	0.229	0.000	0.000	0.979	5.580	1.709	0.043	0.005	0.004
	GraphNorm	0.555	0.826	1.259	0.628	0.725	0.779	1.207	0.590	5.203	0.010	0.324	0.017
	BatchNorm	0.495	1.845	0.477	0.576	0.426	0.832	0.785	0.556	3.851	0.005	0.017	0.018
	ExpreNorm	0.551	0.178	0.148	0.366	0.500	1.001	0.793	0.693	0.503	0.009	0.011	0.001
	MotifNorm	0.811	1.039	2.188	0.382	0.848	0.931	0.991	1.057	0.934	0.011	0.011	0.007
GIN	NoNorm	0.359	2.250	0.766	0.915	0.210	0.667	1.049	4.428	4.058	0.009	0.010	0.073
	GraphNorm	0.908	0.869	1.065	0.514	0.327	0.280	1.059	1.653	1.404	0.012	0.344	0.008
	BatchNorm	0.567	2.302	0.787	0.288	0.723	1.133	1.060	2.830	2.513	0.012	0.012	0.201
	ExpreNorm	0.449	1.747	0.345	0.435	0.730	1.028	1.164	1.745	1.706	0.013	0.013	0.194
	MotifNorm	0.814	1.026	0.379	00.00	0.426	1.003	0.833	1.745	1.734	0.019	0.013	0.163

Table 10: The **standard deviations** of the performance (Table 4) on the node prediction task and link prediction task. We use GCN, GAT and GIN as the basic backbones and set the number of layers as 4, 16 and 32.

	Settings	Pubmed			Citeseer			ogbn-proteins			ogbl-collab		
		4	16	32	4	16	32	4	8	16	4	8	16
GCN	NoNorm	1.127	6.022	2.702	2.106	1.625	5.082	1.692	0.647	0.901	0.501	1.072	1.097
	PairNorm	2.336	6.973	4.299	2.828	2.485	1.169	2.302	0.350	0.440	2.816	1.688	1.250
	NodeNorm	1.147	1.229	1.388	1.677	1.628	0.986	1.459	0.765	0.889	1.011	0.679	2.495
	MeanNorm	0.801	3.586	6.623	1.962	3.546	3.270	1.996	0.384	0.449	1.429	2.345	1.217
	GroupNorm	1.313	5.373	6.069	1.306	2.963	1.630	2.269	0.619	1.414	1.911	2.357	2.244
	BatchNorm	0.690	4.090	3.073	2.057	3.062	2.679	2.140	1.631	1.054	0.271	1.198	1.746
	MotifNorm	1.356	1.110	2.832	2.009	2.879	3.523	1.350	1.875	2.502	1.185	0.443	0.543
GraphSage	NoNorm	0.884	3.525	2.065	1.519	2.444	2.202	4.640	0.571	0.112	2.378	4.161	0.787
	PairNorm	1.662	6.985	4.263	1.525	3.734	1.095	3.381	0.267	0.673	1.252	2.566	2.665
	NodeNorm	1.054	0.000	0.000	1.035	1.751	1.221	3.647	1.191	1.350	2.539	0.051	0.045
	MeanNorm	0.914	1.450	6.857	1.402	3.695	2.876	3.764	6.502	8.403	2.775	2.204	1.404
	GroupNorm	1.058	4.585	1.515	1.560	3.353	3.483	2.606	1.720	0.169	1.868	2.445	1.433
	BatchNorm	1.717	4.636	3.461	2.969	2.792	1.675	3.251	3.028	0.498	1.426	4.112	1.580
	MotifNorm	1.029	2.092	2.487	1.641	2.055	2.211	1.827	3.006	3.375	0.673	0.269	0.237