Non-IID Graph Neural Networks

Yiqi Wang*
Michigan State University
wangy206@msu.edu

Yao Ma*
Michigan State University
mayao4@msu.edu

Charu Aggarwal
IBM T. J. Watson Research Center
charu@us.ibm.com

Jiliang Tang
Michigan State University
tangjili@msu.edu

ABSTRACT

Recently Graph Neural Networks (GNNs) have greatly advanced the task of graph classification. When building a GNN model for graph classification, the graphs in the training set are often assumed to be identically distributed. However, these graphs could have dramatically distinct structures, which indicates that these graphs could be non-identically distributed. Therefore, in this paper, we aim to develop graph neural networks for graphs that are assumed to be not non-identically distributed. Specifically, we propose a general non-IID graph neural network framework, i.e., Non-IID-GNN. Given a graph, Non-IID-GNN can adapt any existing graph neural network model to generate a sample-specific model for this graph. Comprehensive experiments on various graph classification benchmarks demonstrate the effectiveness of the proposed framework. We will release the implementation of the proposed framework upon the acceptance of the paper.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

graph classification, graph neural networks, non-identically distributed

ACM Reference Format:
Yiqi Wang, Yao Ma, Charu Aggarwal, and Jiliang Tang. 2018. Non-IID Graph Neural Networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Graphs are natural representations for many real-world data such as social networks [8, 10, 21, 22], biological networks [1, 4, 16, 18] and chemical molecules [2, 5, 7]. A crucial step to perform downstream tasks on graph data is to learn better representations. Deep neural networks have demonstrated great capabilities in representation learning for Euclidean data and thus have advanced numerous fields including speech recognition [13], computer vision [9] and natural language processing [3]. However, they cannot be directly applied to graph data due to its complex topological structure. Recently, Graph Neural Networks (GNNs) have generalized deep neural networks to graph data that typically perform transforming, propagating and aggregating node features across the graph. They have boosted the performance of many graph related tasks such as node classification [8, 10], link prediction [6, 17, 25] and graph classification [11, 23]. In this work, we aim to advance Graph Neural Networks for graph classification.

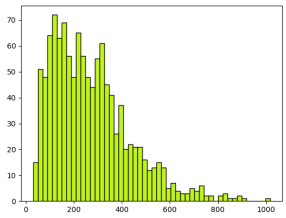
In graph classification, each graph is treated as a data sample and the goal is to train a classification model on a set of training graphs that can predict the label for an unlabeled graph by leveraging its associated node features and graph structure. There are numerous real-world applications for graph classification. For example, it can be used to infer whether a protein functions as an enzyme or not where proteins are denoted as graphs [4]; and it can be applied to forecast Alzheimer's disease progression in which individual brains are represented as graphs [19]. In reality, graphs in the same training set can present distinct structural information. Figure 1a demonstrates the distribution of the number of nodes for protein graphs in the D&D dataset [4] where the number of nodes varies dramatically from 30 to 5,748. We further illustrate two graphs from D&D in Figures 1b and 1c, respectively. These two graphs present very different structural information such as the number of edges, density and diameters. The above investigations indicate that graphs in the same training set could follow different distributions. In other words, they may be non-identically distributed. In fact, this observation is consistent with existing work. For example, it is evident in [20] that due to differences in individual brains, the distribution of the brain data can vary remarkably across individuals. It naturally raises the question – whether we should treat these training graphs differently? To investigate this question, we divide graphs from D&D into two groups based on the number of nodes – one for graphs with a small number of nodes and the other for graphs with a large number of nodes. Then, we split each group into a training set and a test set. We train two GCN models[1] [10] based on two training sets, separately, and test their performance on the two test sets. The results are shown in the Figure 1d. The GNN model achieves much better performance on the test from the

*Both authors contributed equally to this research.

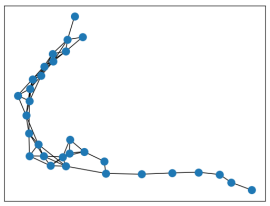
Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

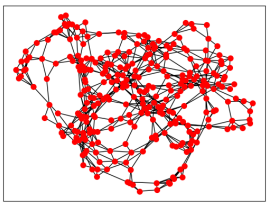
[1]The GCN model was originally designed for semi-supervised node classification, we include a max-pooling layer to generate graph-level representation for graph classification.



(a) Node size distribution



(b) A graph with 31 nodes



(c) A graph with 302 nodes

	small test set	large test set
small training set	0.662	0.557
large training set	0.472	0.778

(d) Classification accuracy

Figure 1: An Illustrative Example of Varied Structural Information and its Impact on the Performance of Graph Neural Network based Graph Classification.

same group that suggests that efforts are desired to consider the difference.

In this paper, we propose to design graph neural networks for graphs that are assumed to be non-identically distributed. In particular, we target on addressing two challenges – (a) how to capture the distributions of graphs that are often not available; and (b) how to integrate them to build graph neural networks for graph classification. To tackle these two challenges, we propose a novel graph neural network framework, Non-IID-GNN, for graph classification, which can learn graph-level representations for non-identically distributed graphs. We design comprehensive experiments on numerous graph datasets from various domains to verify the effectiveness of the proposed framework.

2 THE PROPOSED FRAMEWORK

The majority of traditional graph neural networks assume that graphs in the same training data are identically distributed and thus they train a unified GNN model for all graphs. In this section, we introduce the proposed framework Non-IID-GNN that has been designed for graphs assumed to be non-identically distributed.

2.1 The Overall Design

In this work, graphs are assumed to be non-identically distributed. Thus, we are desired to build distinct GNN models for graphs with different distributions. To achieve this goal, we face tremendous challenges. First, we have no explicit knowledge about the underlying distributions of graphs. Second, if we separately train different models for graphs with different distributions, we have to split the training graphs for each model; as a consequence, the training data for each model could be very limited. For example, in the extreme case when one graph has a unique distribution, we only have one training sample for the corresponding model. Third, even if we can well train distinct GNN models for different graphs, during the test stage, for an unlabelled graph, which trained model we should adopt to make the prediction?

In this work, we propose a Non-IID graph neural network framework, i.e., Non-IID-GNN, which can tackle the aforementioned challenges simultaneously. An overview about the architecture of Non-IID-GNN is demonstrated in Figure 2. The basic idea of Non-IID-GNN is – it approximates the distribution information of a graph sample g_i via an adaptor network on its structural information, which serves as the adaptor parameters to adapt each GNN block for g_i and the adapted GNN model GNN_i can be viewed as a

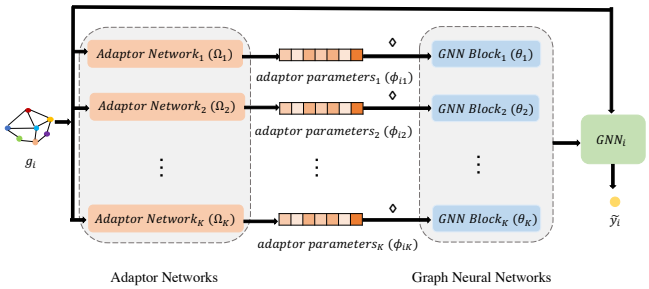


Figure 2: An overview of the proposed Non-IID graph neural network.

specific graph classification model for g_i. The underlying distribution of a given sample may have different influences on different GNN blocks. Thus, for each GNN block, we introduce one adaptor network. To solve the first challenge of no knowledge about the underlying distribution, we develop an adaptor network to approximate the distribution information of a graph through its observed structural information. To tackle the second challenge, we learn a set of shared models including adaptor networks and GNN blocks, which are trained among all the graph samples, and thus preseving the common knowledge from the whloe dataset. With this design, the third challenge is addressed automatically. Given an unlabeled graph g_j, the trained Non-IID-GNN will generate an adapted GNN model GNN_j to predict its label. Next we will introduce details about the adaptor network, the adapted graph neural network for each graph, and the time complexity analysis.

2.2 The Adaptor Network

The goal of the adaptor network is to approximate the distribution information of a given graph. In particular, we utilize the structural information as input for the adaptor network to achieve this goal. The intuition is – the structural differences of graphs are caused by their different distributions; thus, we want to estimate the distribution from the observed structural information via a powerful adaptor network. Graph neural networks often consist of several subsequent filtering and pooling layers, which can be viewed as different blocks of the graph neural network model. As mentioned before, the distribution of a graph may influence each GNN block differently. Thus, we build an adaptor network to generate adaptor parameters for each block. We first extract a vector $\mathbf{s}_i$ to denote the structural information of a given graph g_i. We will discuss more details about $\mathbf{s}_i$ in the experiment section. As shown in the left part

of Figure 2, the adaptor networks take the structural information $\mathbf{s}_i$ as input and generate the adaptation parameters for each block. Assuming that we have K blocks in the graph neural network, we have K independent adaptor networks. Note that these adaptor networks share the same input $\mathbf{s}_i$ while their outputs can be different. Specifically, the adaptor network for the j-th block can be expressed as follows:

$$\boldsymbol{\phi}_{ij} = h_j(\mathbf{s}_i; \Omega_j), j = 1, \ldots, K, \tag{1}$$

where Ω_j denotes the parameters of the j-th adaptor network and $\boldsymbol{\phi}_{ij}$ denotes its output, which will be used to adapt the j-th learning block. The adaptor $\mathbf{h}_j$ can be modeled using any functions. In this work, we utilize feed-forward neural networks due to their strong capability in approximating any functions. For convenience, we summarize the process of the K adaptor networks for g_i as follows:

$$\Phi_i = H(\mathbf{s}_i; \Omega_H), \tag{2}$$

where Φ_i contains the generated adaptation parameters of the graph g_i for all the GNN blocks and Ω_H denotes the parameters of the K adaptor networks.

2.3 The Adapted Graph Neural Network

Any existing graph neural network model can be adapted by the Non-IID-GNN framework to generate sample-specific models based on the sample's structural information. Therefore, we first generally introduce the GNN model for graph classification and describe how it can be adapted based on a specific given sample. Then, we illustrate how to adapt a specific GNN model.

2.3.1 A General Adapted Framework. A typical GNN framework for graph classification usually contains two types of layers, i.e., the filtering layer and the pooling layer. The filtering layer takes the graph structure and node representations as input and generates refined node representations as output. The pooling layer takes graph structure and node representations as input to produce a coarsened graph with a new graph and new node representations. A general GNN framework for graph classification contains K_p pooling layers, each of which follows K_f stacking filtering layers. Hence, there are $K = K_p * K_f$ learning blocks in this GNN framework. A graph-level representation can be obtained from these layers that can be further utilized to perform the prediction. Given a graph sample g_j, we need to adapt each of the K layers according to its distribution information from the adaptor network. Via this process, we can generate a GNN model GNN_j specific to g_j.

Without the loss of generality, when introducing a filtering layer or a pooling layer, we use an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and node representations $\mathbf{X} \in \mathbb{R}^{n \times d}$ to denote the input of these layers where n is the number of nodes and d is the dimension of node features. Then, the operation of a filtering layer can be described as follows:

$$\mathbf{X}_{new} = f(\mathbf{A}, \mathbf{X}; \theta_f) \tag{3}$$

where θ_f denotes the parameters in the filtering layer and $\mathbf{X}_{new} \in \mathbb{R}^{n \times d_{new}}$ denotes the refined node representations with dimension d_{new} generated by the filtering layer. Assuming ϕ_f is the corresponding adaptor parameters for this filtering layer, we adapt the model parameter θ_f of this filtering layer as follows:

$$\theta_f^m = \theta_f \diamond \phi_f, \tag{4}$$

where θ_f^m is the adapted model parameters that has the same dimension as the original model parameter θ_f; and $\diamond$ is the adaptation operator. The adaption operator can have various designs, which can be determined according to the specific GNN model. We will provide the details of the adaptation operator when we introduce concrete examples in the following subsections. Then, with the adapted model parameters, we can define the adapted filtering layer as follows:

$$\mathbf{X}_{new} = f(\mathbf{A}, \mathbf{X}; \theta_f \diamond \phi_f). \tag{5}$$

The process of a pooling layer can be described as follows:

$$\mathbf{A}_{new}, \mathbf{X}_{new} = p(\mathbf{A}, \mathbf{X}; \theta_p), \tag{6}$$

where θ_p denotes the parameters of the pooling layer, $\mathbf{A}_{new} \in \mathbb{R}^{n_{new} \times n_{new}}$ with $n_{new} < n$ is the adjacency matrix for the newly generated coarsened graph and $\mathbf{X}_{new} \in \mathbb{R}^{n_{new} \times d_{new}}$ is the learned node representations for the coarsened graph. Similarly, we adapt the model parameters of the pooling layer as follows:

$$\theta_p^m = \theta_p \diamond \phi_p, \tag{7}$$

which leads to the following adapted pooling layer:

$$\mathbf{A}_{new}, \mathbf{X}_{new} = p(\mathbf{A}, \mathbf{X}; \theta_p \diamond \phi_p), \tag{8}$$

where ϕ_p is the adaptation parameters generated by the adaptor network for this pooling layer.

To summarize, given a graph sample g_i, its specific adaptor parameters Φ_i learned by the adaptor networks, and a GNN framework $GNN(\cdot|\Theta_{GNN})$ with model parameters of all layers summarized in Θ_{GNN}, we can generate an adapted GNN specific for the sample g_i as $GNN_i(\cdot|\Theta_{GNN} \diamondsuit \Phi_i)$. Here, we summarize the layer-wise adaption operation using $\Theta_{GNN} \diamondsuit \Phi_i$. There are numerous GNN models designed for graph classification [6, 11, 15, 24]. The proposed framework can be applied to the majority of these models. In this work, we focus on two representative GNN models including GCN [10] and Diffpool [23]. We would like to leave the investigations of other GNN models as one future work.

2.3.2 Adapted GCN: Non-IID-GCN. Graph Convolutional Network (GCN) [10] is originally proposed for semi-supervised node classification task. The filtering layer in GCN is defined as follows:

$$\mathbf{X}_{new} = f(\mathbf{A}, \mathbf{X}; \theta_f) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X} \mathbf{W}), \tag{9}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ represents the adjacency matrix with self-loops, $\tilde{\mathbf{D}} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$ and $\mathbf{W} \in \mathbb{R}^{d \times d_{new}}$ denotes the trainable weight matrix in filtering layer and $\sigma(\cdot)$ is a nonlinear activation function. With the adaptation parameter ϕ_f for the corresponding filtering layer, the adapted filtering layer can be represented as follows:

$$\mathbf{X}_{new} = f(\mathbf{A}, \mathbf{X}; \theta_f) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X} (\mathbf{W} \diamond \phi_f)). \tag{10}$$

Specifically, we adopt FiLM [14] as the adaption operator. In this case, the dimension of the adaptor parameter is $2d$, i.e., $\phi_f \in \mathbb{R}^{2d}$. We split ϕ_f into two parts $\gamma_f \in \mathbb{R}^d$ and $\beta_f \in \mathbb{R}^d$ and then the adaptation operation can be expressed as follows

$$\mathbf{W} \diamond \phi_f = (\mathbf{W} \odot br(\gamma_f, d_{new})) + br(\beta_f, d_{new}), \tag{11}$$

where $br(\mathbf{a}, k)$ is a broadcasting function that repeats k times for the vector $\mathbf{a}$; hence, $br(\gamma_f, d_{new}) \in \mathbb{R}^{d \times d_{new}}$ and $br(\beta_f, d_{new}) \in$

$\mathbb{R}^{d \times d_{new}}$ have the same shape as $\mathbf{W}$ and $\odot$ denotes the element-wise multiplication between two matrices.

To utilize GCN for graph classification, we introduce a node-wise max pooling layer to generate graph representation from the node representations as follows:

$$\mathbf{x}_G = p(\mathbf{A}, \mathbf{X}; \theta_p) = max(\mathbf{X}), \tag{12}$$

where $\mathbf{x}_G \in \mathbb{R}^{d_{new}}$ denotes the graph-level representation and $max()$ takes the maximum over all the nodes. Note that the max-pooing operation does not involve learable parameters and thus no adaptation is needed for it. We refer to an adapted GCN framework as Non-IID-GCN.

2.3.3 Adapted diffpool: Non-IID-Diffpool. Diffpool is a hierarchical graph level representation learning method for graph classification [23]. The filtering layer in Diffpool is the same as (9) and its corresponding adapted version is shown in (10). Its pooling layer is defined as follows:

$$\mathbf{S} = softmax(f_a(\mathbf{A}, \mathbf{X}; \theta_{f_a})), \tag{13}$$

$$\mathbf{X}_{new} = \mathbf{S}^T\mathbf{Z}, \tag{14}$$

$$\mathbf{A}_{new} = \mathbf{S}^T\mathbf{A}\mathbf{S}, \tag{15}$$

where f_a is a filtering layer embedded in the pooling layer, $\mathbf{S} \in \mathbb{R}^{n \times n_{new}}$ is a soft-assignment matrix, which softly assigns each node into a supernode to generate a coarsened graph. Specifically, the structure and the node representations for the coarsened graph are generated by (15) and (14) respectively, where $\mathbf{Z} \in \mathbb{R}^{n \times d_{new}}$ is the output of the filtering layers to the pooling layer. To adapt the pooling layer, we only need to adapt (13), which follows the same way as introduced in (10) as it is also a filtering layer. We refer to the adapted diffpool model as Non-IID-Diffpool.

3 EXPERIMENT

In this section, we conducted comprehensive experiments to verify the effectiveness of the proposed Non-IID-GNN framework. We first describe the implementation details of the proposed framework. Then, we evaluate the performance of the framework by comparing original GCN and Diffpool with the adpated GCN, Diffpool models by the Non-IID-GNN framework. Next, we analyse the importance of different components in the adaptor operator. Finally we conduct some case studies to further facilitate our understanding of the proposed method.

3.1 Experimental Settings

We carried out graph classification tasks on eight datasets from various domains with a variety of representative baselines. Next, we describe the datasets and the baselines.

We choose eight graph datasets to evaluate the proposed framework including **D&D** [4], **ENZ** [18], **PROT** [1], **NCI1** and **NCI109** [18], **COLLAB**, **RE-BI** and **RE-5K** [22]. More details about these datasets are demonstrated in Table 1.

In Section 2.3, we apply the proposed framework to two graph neural networks: a basic graph convolutional network (GCN) [10] and an advanced graph convolutional network with hierarchical pooling, Diffpool [23]. The corresponding adapted versions are Non-IID GCN and Non-IID Diffpool, respectively. *Our evaluation purpose is if the proposed framework can boost the performance of existing models by adapting them to their corresponding NonIID versions.* Thus, (1) to validate the effectiveness of the proposed model, we compare Non-IID-GCN, Non-IID-Diffpool with GCN and Diffpool; and (2) we do not choose models in [6, 11, 15, 24] as baselines since the proposed framework can be applied to adapt them as well. Besides, we also develop baseline methods , Multi-GCN and Multi-Diff. They learn multiple graph convolutional networks for graph samples with different structural information. **Multi-GCN (or Multi-Diff)** consists of several GCN (or Diffpool) models trained from different subsets of the training dataset. We first cluster data samples from training set into different training subsets based on the graph structural information. *Note that in this work, the structural information $\mathbf{s}_i$ of g_i includes the number of nodes, the number of edges and the graph density.* Then we train different models from different training subsets. During the test phase, given a test graph sample, we first assign it to one cluster with the smallest euclidean distance between its graph structural information and the centroid of the training cluster. Then, we choose the model trained on the cluster for prediction. In this experiment, we set the number of clusters to 2 and 3, and denote the corresponding frameworks as Multi-GCN-2 (or Multi-Diff-2) and Multi-GCN-3 (or Multi-Diff-3).

Table 1: The statistics of eight datasets. #Graphs denotes the number of graphs. #Class denotes the number of graph classes. #Nodes(avg ± std) denotes the average and standard deviation of the number of nodes among the graphs.

Datasets	#Graphs	#Class	#Nodes(avg± std)
DD	1,178	2	284.3± 272.0
ENZ	600	6	32.6± 14.9
PROT	1,113	2	39.06± 45.8
NCI1	4,110	2	29.87± 13.5
NCI109	4,127	2	29.68± 13.6
COLLAB	5,000	3	74.49± 62.3
RE-BI	2,000	2	429.63± 554.0
RE-5K	4,999	5	508.52± 452.6

3.2 Graph Classification Performance Comparison

For each graph dataset, we randomly shuffle the dataset and then split 90% of the data into the training set and the remaining 10% as test set. We train all the models on the training set and evaluate their performance on the test set with accuracy as the measure. We repeat this process for 10 times and report the average performance. The GCN/Non-IID-GCN model consists of 3 filtering layers and a single max-poling layer; the hidden dimension of each filtering layer is 20; and ReLU [12] activation is applied after each filtering layer. For Diffpool/Non-IID-Diffpool, we follow the setting of the original paper [23] with $K_p = 2$, $K_f = 3$ and the dimension of hidden filtering layer 20. We adopt fully-connected networks to implement the adaptor networks in the Non-IID-GNN frameworks. Its input dimension is the same as the dimension of the graph structural information.

Table 2: Comparisons of graph classification performance of in terms of accuracy.

Methods	Datasets							
	DD	ENZ	PROT	NCI1	NCI109	COLLAB	RE-BI	RE-5K
GCN	0.7716	0.5176	0.7662	0.7715	0.7574	0.6986	0.8189	0.5039
Diffpool	0.7823	0.5771	0.7894	0.8017	0.7718	0.704	0.8972	0.5646
Multi-GCN-2	0.7435	0.4521	0.7950	0.7743	0.7515	0.699	0.7938	0.5088
Multi-GCN-3	0.7435	0.4604	0.7962	0.7749	0.7555	0.6815	0.8888	0.470
Multi-Diff-2	0.7672	0.5292	0.8001	0.7948	0.7797	0.7168	0.8736	0.5385
Multi-Diff-3	0.7716	0.4896	0.8255	0.7908	0.7797	0.7178	0.8896	0.5312
Non-IID-GCN	0.7931	0.5592	0.7788	0.7877	0.7705	0.7316	0.9039	0.5293
Non-IID-Diffpool	0.7856	0.5854	0.7939	0.7932	0.7755	0.738	0.9292	0.5541

Table 3: Ablation Study

Methods	Datasets							
	DD	ENZ	PROT	NCI1	NCI109	COLLAB	RE-BI	RE-5K
GCN	0.7716	0.5176	0.7662	0.7715	0.7574	0.6986	0.8189	0.5039
Non-IID-GCN$_\gamma$	0.781	0.5225	0.7761	0.779	0.7596	0.7084	0.8517	0.5173
Non-IID-GCN$_\beta$	0.7797	0.54	0.7793	0.7877	0.7714	0.7116	0.8878	0.5188
Non-IID-GCN	0.7931	0.5592	0.7788	0.7878	0.7705	0.7316	0.9039	0.5293

Table 4: Adaptability Study

Datasets	Methods			
	Non-IID-Diffpool	Diffpool	Non-IID-GCN	GCN
ENZ	0.2564	0.2222	0.2222	0.2051
RE-BI	0.7855	0.5265	0.7019	0.5042

The results are shown in Table 2. We notice that Multi-GCN and Multi-Diff frameworks sometimes can obtain comparable performance with their original versions. This observation suggests that training graphs should be considered differently. However, most of the time, Non-IID-GCN and Non-IID-Diffpool outperform the corresponding Multi-GCN and Multi-Diff frameworks. Simply training different models for different graphs can lead to unsatisfactory performance because less training data is available for each model. We observe that the adapted GCN model, Non-IID-GCN, consistently obtains better performance than the original GCN model. We also find similar observations when comparing Non-IID-Diffpool with the original Diffpool model. These observations demonstrate that the sample-wise adaptation performed by the Non-IID-GNN framework can boost the performance of GNN frameworks.

To further show the adaptability of the proposed framework to new graphs with different distributions, we first order graphs according to their sizes from small to big and then we choose first 80% as training and the remaining 20% as test. The purpose of this setting is to simulate the distributions of graphs in the test are different from these in the training set. The results on the ENZ and RE-BI datasets are shown in Table 4. We can observe that the advantage of the non-iid frameworks is much more significant under this setting that demonstrates the ability to adapt to new graphs.

3.3 Ablation Study

In this subsection, we investigate the effectiveness of different components in the adaptor operator in Equation (11) used in our model. Specifically, we want to investigate whether γ_f and β_f play important roles in the adaptor operator by defining the variants of Non-IID-GCN –**Non-IID-GCN**$_\gamma$: It is a variant of the adaptor operator with only element-wise multiplication operation where instead of (11), the adaptation process is now expressed as: $\mathbf{W} \diamond \phi_f = (\mathbf{W} \odot br(\gamma_f, d_{new}))$; and **Non-IID-GCN**$_\beta$: It is a variant of the adaptor operator with only element-wise addition operation where instead of (11), the adaptation process is now: $\mathbf{W} \diamond \phi_f = \mathbf{W} + br(\beta_f, d_{new})$.

Following the previous experimental setting, we compared Non-IID-GCN with its variants. The results are presented in Table 3. We observe that both **Non-IID-GCN**$_\gamma$ and **Non-IID-GCN**$_\beta$ can outperform the original GCN model. It indicates that both terms with γ and β are effective for the adaptation and utilizing either one of them can already adapt the original model in a reasonable manner. We also note that the Non-IID-GCN model outperforms both **Non-IID-GCN**$_\gamma$ and **Non-IID-GCN**$_\beta$ on most of the datasets. It demonstrates that the adaption effects of the term with γ and β are complementary to each other and combing them together can further enhance the performance.

4 CONCLUSION

In this paper, we propose a general graph neural network framework, Non-IID-GNN, to deal with graphs that are non-identically distributed. Given a graph sample, the Non-IID-GNN framework is able to approximate its underlying distribution information from its structural information, the Non-IID-GNN framework can then adapt any existing GNN-based graph classification model to generate a specific model for this sample, which is then utilized to predict the label of this sample. Comprehensive experiments demonstrated that the Non-IID-GNN framework can effectively adapt both flat

GNN model and hierarchical GNN model to enhance their performance. An interesting future direction is to better infer the underlying distribution given a graph sample. Instead of utilizing hand-engineered graph properties to approximate the underlying distribution information of a given sample, we can design more sophisticated algorithm to achieve this goal.

5 ACKNOWLEGEMENTS

This work is supported by the National Science Foundation (NSF) under grant numbers IIS-1714741, IIS-1715940, IIS-1845081, IIS-1907704, IIS-1928278 and CNS-1815636.

REFERENCES

[1] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl_1 (2005), i47–i56.

[2] Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*. 2702–2711.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*. 2224–2232.

[6] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. *arXiv preprint arXiv:1905.05178* (2019).

[7] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1263–1272.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

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

[11] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. 2019. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 723–731.

[12] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[13] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. 2019. Speech recognition using deep neural networks: A systematic review. *IEEE Access* 7 (2019), 19143–19165.

[14] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[15] Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. 2019. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. *arXiv preprint arXiv:1911.07979* (2019).

[16] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. 2004. BRENDA, the enzyme database: updates and major new developments. *Nucleic acids research* 32, suppl_1 (2004), D431–D433.

[17] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. 2017. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in neural information processing systems*. 991–1001.

[18] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, Sep (2011), 2539–2561.

[19] Tzu-An Song, Samadrita Roy Chowdhury, Fan Yang, Heidi Jacobs, Georges El Fakhri, Quanzheng Li, Keith Johnson, and Joyita Dutta. 2019. Graph Convolutional Neural Networks For Alzheimer's Disease Classification. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 414–417.

[20] Robert E Tillman. 2009. Structure learning with independent non-identically distributed data. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1041–1048.

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

[22] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1365–1374.

[23] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*. 4800–4810.

[24] Hao Yuan and Shuiwang Ji. 2020. StructPool: Structured Graph Pooling via Conditional Random Fields. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJxg_hVtwH

[25] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks.. In *IJCAI*, Vol. 18. 3155–3161.