Non-IID Graph Neural Networks

Yiqi Wang∗
Michigan State University
wangy206@msu.edu

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

Yao Ma∗
Michigan State University
mayao4@msu.edu

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:

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[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
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 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 preserving the common knowledge from the whole 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 $s_i$ to denote the structural information of a given graph $g_i$. We will discuss more details about $s_i$ in the experiment section. As shown in the left part
of Figure 2, the adaptor networks take the structural information \( s_i \) 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 \( s_i \) while their outputs can be different. Specifically, the adaptor network for the \( j \)-th block can be expressed as follows:

\[
\phi_{ij} = h_j(s_i; \Omega_j), \quad j = 1, \ldots, K,
\]

where \( \Omega_j \) denotes the parameters of the \( j \)-th adaptor network and \( \phi_{ij} \) denotes its output, which will be used to adapt the \( j \)-th learning block. The adaptor \( h_j \) 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:

\[
\Phi_i = H(s_i; \Omega_H),
\]

where \( \Phi_i \) contains the generated adaptation parameters for this filtering layer, we adapt the corresponding adaptor parameters for this filtering layer, we adapt the model parameter \( \theta_f \) of this filtering layer as follows:

\[
\theta_f^m = \theta_f \circ \phi_f,
\]

where \( \theta_f^m \) is the adapted model parameters that has the same dimension as the original model parameter \( \theta_f \) and \( \circ \) is the adaption 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 adaption 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:

\[
X_{new} = f(A, X; \theta_f \circ \phi_f).
\]

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

\[
A_{new}, X_{new} = p(A, X; \theta_p),
\]

where \( \theta_p \) denotes the parameters of the pooling layer, \( A_{new} \in \mathbb{R}^{n_{new} \times n_{new}} \) with \( n_{new} < n \) is the adjacency matrix for the newly generated coarsened graph and \( X_{new} \in \mathbb{R}^{n_{new} \times d_{new}} \) is the learned node representations for the coarsened graph. Similarly, we adapt the model parameters of the pooling layer as follows:

\[
\theta_p^m = \theta_p \circ \phi_p,
\]

which leads to the following adapted pooling layer:

\[
A_{new}, X_{new} = p(A, X; \theta_p \circ \phi_p),
\]

where \( \phi_p \) is the adaptation parameters generated by the adaptor network for this pooling layer.

To summarize, given a graph sample \( g_i \), its specific adaption parameters \( \Phi_i \) learned by the adaptor networks, and a GNN framework \( \Phi_i(\cdot| \Omega_{GNN}) \) with model parameters of all layers summarized in \( \Theta_{GNN} \), we can generate an adapted GNN specific for the sample \( g_i \) as \( \Phi_i(\cdot| \Omega_{GNN} \circ \Phi_i) \). Here, we summarize the layer-wise adaption operation using \( \Theta_{GNN} \circ \Phi_i \). 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 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 \( K = K_p \times K_f \) 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 \( g_i \), 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 \( GNN_i \) specific to \( g_i \).

Without the loss of generality, when introducing a filtering layer or a pooling layer, we use an adjacency matrix \( A \in \mathbb{R}^{n \times n} \) and node representations \( X \in \mathbb{R}^{n \times d} \) 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:

\[
X_{new} = f(A, X; \theta_f)
\]

where \( \theta_f \) denotes the parameters in the filtering layer and \( X_{new} \in \mathbb{R}^{n \times d_{new}} \) denotes the refined node representations with dimension \( d_{new} \) generated by the filtering layer. Assuming \( \phi_f \) is the corresponding adaptor parameters for this filtering layer, we adapt the model parameter \( \theta_f \) of this filtering layer as follows:

\[
\theta_f^m = \theta_f \circ \phi_f,
\]
We carried out graph classification tasks on eight datasets from Woodstock ’18, June 03–05, 2018, Woodstock, NY as introduced in (10) as it is also a filtering layer. We refer to \( f \) where \( x_G = p(A, X, \theta_p) = \max(X) \), (12)

where \( x_G \in \mathbb{R}^{d_{new}} \) denotes the graph-level representation and \( \max() \) takes the maximum over all the nodes. Note that the max-pooling operation does not involve learnable 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:

\[
S = \text{softmax}(f_a(A, X, \theta_{f_a})),
\]

(13)

\[
x_{new} = S^T Z,
\]

(14)

\[
A_{new} = S^T A,
\]

(15)

where \( f_a \) is a filtering layer embedded in the pooling layer, \( S \in \mathbb{R}^{n \times n_{new}} \) 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 \( Z \in \mathbb{R}^{n \times d_{new}} \) 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 adapted 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 DD [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 Non-IID 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 \( s_i \) 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

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

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-pooling 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.
We observe that the adapted GCN model, Non-IID-GCN, consistently outperforms the original GCN model. 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.

| Table 2: Comparisons of graph classification performance of in terms of accuracy. |
|-----------------------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Methods | 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 | 0.8097 | 0.7718 | 0.704 | 0.8972 | 0.5646 |
| Multi-GCN-2 | 0.7435 | 0.4521 | 0.7950 | 0.7743 | 0.7515 | 0.6815 | 0.8888 | 0.470 |
| 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 | 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-GCNγ | 0.781 | 0.5225 | 0.7761 | 0.779 | 0.7596 | 0.7084 | 0.8517 | 0.5173 |
| Non-IID-GCNδ | 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 | 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 |

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 and β play important roles in the adaptor operator by defining the variants of Non-IID-GCN – Non-IID-GCNγ and Non-IID-GCNδ. It is a variant of the adaptor operator with only element-wise multiplication operation where instead of (11), the adaption process is now expressed as: \( W \circ \phi_f = (W \otimes \beta) \), and Non-IID-GCNγ is a variant of the adaptor operator with only element-wise addition operation where instead of (11), the adaption process is now: \( W \circ \phi_f = W + br(\gamma, d_{new}) \).

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-GCNγ and Non-IID-GCNδ 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-GCNγ and Non-IID-GCNδ 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 given a graph sample, we can design more sophisticated algorithm to achieve this goal.

5 ACKNOWLEDGEMENTS

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


