Journal of Machine Learning Research 23 (2022) 1-26 Submitted 1/21; Revised 5/22; Published 9/22

Stable Prediction on Graphs with Agnostic Distribution Shifts

Shengyu Zhang sy_zhang@zju.edu.cn
College of Computer Science and Technology, Zhejiang University,
Hangzhou 310027, China

Yunze Tong tyz01@zju.edu.cn
College of Computer Science and Technology, Zhejiang University,
Hangzhou 310027, China

Kun Kuang* kunkuang@zju.edu.cn
College of Computer Science and Technology, Zhejiang University
Hangzhou 310027, China

Fuli Feng fulifeng93@gmail.com
University of Science and Technology of China
Hefei 230026, China

Jiezhong Qiu qiujz16@mails.tsinghua.edu.cn
Department of Computer Science and Technology, Tsinghua University
Beijing 100084, China

Jin Yu yujin.yujin@gmail.com
Alibaba Group
Hangzhou 310052, China

Zhou Zhao zhouzhao@zju.edu.cn
College of Computer Science and Technology, Zhejiang University
Hangzhou 310027, China

Hongxia Yang yang.yhx@alibaba-inc.com
Alibaba Group
Beijing 100084, China

Zhongfei Zhang zzhang@binghamton.edu
Department of Computer Science Watson School of Engineering and Applied Sciences, Binghamton University
New York, United States 13850

Fei Wu* wufei@zju.edu.cn
College of Computer Science and Technology, Zhejiang University
Hangzhou 310027, China *

Editor: My editor

Abstract

Most graph neural networks (GNNs) are proposed and evaluated under independent and identically distributed (IID) training and testing data. In real-world applications, however,

* Corresponding authors

agnostic distribution shifts from training to testing naturally exist, leading to unstable prediction of traditional GNNs. To bridge the gap, we pursue stable prediction on graphs, *i.e.*, to achieve high average performance and low performance variance (stability) across non-IID testing graphs. The key to stable prediction lies in capturing stable properties that are resilient to distribution shifts. In this light, we aim to identify neighbor nodes (properties) in neighborhood aggregation that are consistently important for prediction under heterogeneous distribution shifts. To achieve this target, we propose a model-agnostic stable learning framework for GNNs. The framework performs biased selection on the observed training graph, resulting in multiple non-IID graph subsets. We train one weight predictor per subset to measure the importance of properties under a particular distribution shift, and multiple predictors could tell the properties that are consistently important. An important property should contribute to high average performance and also stability (low performance variance) across non-IID subsets. In this regard, in training importance predictors, we introduce a globally stable regularizer to reduce the variance of training losses across non-IID graph datasets. Based on the importance weights of properties across non-IID subsets, a locally stable regularizer down-weights unstable properties in prediction. We conduct extensive experiments on several graph benchmarks and a noisy industrial recommendation dataset where distribution shifts exist. The results demonstrate that our method outperforms various state-of-the-art GNNs for stable prediction on graphs with agnostic distribution shifts.

Keywords: Stability; Stable Prediction; Graph Neural Network; Distribution Shift

1 Introduction

Graphs are commonly used to represent the complex structures of interacting entities in a flexible manner (Sankar et al., 2021; Park et al., 2019; Wang et al., 2019c). Recent advances in the literature have convincingly demonstrated the high capability of Graph Neural Networks (GNNs) (Kipf and Welling, 2017; Velickovic et al., 2018; Scarselli et al., 2009) on graph representation and reasoning. Technically, GNNs follow a neighborhood aggregation scheme, where nodes on the graph are represented by recursively aggregating the information of neighbor nodes (Xu et al., 2018; Gilmer et al., 2017). As such, predictions for nodes are made based on the information of the nodes themselves and their neighbor nodes.

Most GNNs are proposed and evaluated with an underlying assumption, *i.e.*, the training and testing graph data are independent and identically distributed (IID) (Kipf and Welling, 2017; Xu et al., 2019). Unfortunately, the IID assumption might not be satisfied in real-world applications (Hu et al., 2020a; Lohr, 2009). For example, in a paper citation network such as ogbn-arxiv (Hu et al., 2020a), early works that cite the LSTM (Hochreiter and Schmidhuber, 1997) paper would mostly belong to natural language processing (NLP). Due to broadened impact, papers recently citing the LSTM paper belong to a variety of subjects, which means the label distribution of neighbor nodes for the LSTM paper has shifted from training to testing. The distribution shifts might render the predictions of traditional GNNs unstable across non-IID testing datasets. For example, GNNs trained on the past citation network might correlate paper subject (*i.e.*, node label) NLP with the paper citation (*i.e.*, node neighbor) LSTM, leading to false subject predictions for papers citing LSTM on the testing graph. Therefore, learning stable GNNs that are resilient to distribution shifts and make stable predictions on graphs will be important for real-world applications.

Traditional stable learning techniques (Bickel et al., 2009; Dudík et al., 2005; Huang et al., 2006; Liu and Ziebart, 2014; Shimodaira, 2000) typically re-weight training samples to drive the training distribution closer to the testing distribution. These works require prior knowledge of testing distribution while we mostly face agnostic distribution shifts on graphs. Recent works (Kuang et al., 2020, 2018; Shen et al., 2020) proposed novel sample re-weighting techniques for variable decorrelation such that the causation between the outcome and the variables can be recovered for stable prediction. However, they assume either binary predictions or linear models, which are hardly satisfied in graph domains.

In this paper, we aim to learn GNNs that can make stable predictions under agnostic distribution shifts. Following (Kuang et al., 2018, 2020), we consider two quantitative goals for stable prediction on graphs, *i.e.*, high average performance and low performance variance across multiple non-IID testing graphs. The key to stable prediction lies in capturing stable properties, according to (Kuang et al., 2018, 2020). For example, in the paper citation network, papers citing a graph dataset/survey paper (*e.g.*, Open Graph Benchmark (Hu et al., 2020a)) would mostly belong to the graph subject. Such correlation is stable across time, and the cited dataset/survey paper is a stable property of the target paper for subject prediction. To capture stable properties on graphs, we need to answer two critical questions, *i.e.*, Q1) what are the properties of nodes on graphs; and Q2) what kinds of properties are stable? As for Q1, recall that GNNs follow a neighborhood aggregation schema, which means the prediction for a node is made based on the information of the node itself and neighbor nodes. We regard neighbor nodes as the properties of the target node. As for Q2, inspired by (Peters et al., 2016), we propose to take neighbor nodes that are consistently important under heterogeneous distribution shifts as stable properties.

To capture stable neighbor nodes for prediction, we introduce a stable learning framework for GNNs. The framework simulates heterogenous distribution shifts via biased node selection on the observed graph dataset *w.r.t.* node labels and node attributes, resulting in multiple non-IID graph datasets. We employ one importance weight predictor per dataset to measure the importance of properties under a certain distribution shift. It is essential that the importance weight predictor learns the desired importance, which refers to the contribution to the overall performance (task-specific objective), and the stability (low performance variance) across distribution shifts. To achieve this, in training weight predictors, we introduce a globally stable regularizer to explicitly reduce the variance of training losses of non-IID graphs. Note that loss is one measurement of model performance. Based on the importance weights of properties across non-IID subsets, we introduce a locally stable regularizer, which down-weight properties that are not consistently important in prediction.

We conduct experiments on various public graph benchmarks and a noisy real-world recommendation dataset collected during an annual product-promotion festival[1]. The data distribution on the user-item graph shifts due to the change of sale promotion strategies across time. We consider both traditional task-specific evaluation metrics and protocols from the literature of stable learning (Kuang et al., 2018, 2020). Extensive results demonstrate the rationality of our framework on learning GNNs that make stable predictions on graphs. In summary, the contributions of this paper are:

[1]We will release the desensitized dataset to promote further investigations.

- We analyze the distribution shift problem on graphs, and propose to capture stable neighbor nodes (properties) for stable prediction on graphs.
- We devise a stable learning framework for GNNs where the locally stable regularizer and the globally stable regularizer jointly capture stable properties from constructed non-IID training graph datasets.
- We conduct extensive experiments on public graph benchmarks and a noisy industrial recommendation dataset, demonstrating the rationality of the proposed framework.

2 Related Works

The recent advances in graph neural networks (Wu et al., 2019; Klicpera et al., 2019; Velickovic et al., 2018; Kipf and Welling, 2017; Jia et al., 2020; Liu et al., 2020; Zhang et al., 2019; Bojchevski et al., 2020; Wang et al., 2020b) have convincingly demonstrated high capability in capturing structural and relational patterns within graphs. Typically, GNNs follow a message-passing schema for representation learning by transforming and aggregating the information from other nodes within the neighborhood. Different neighborhood aggregation variants (Scarselli et al., 2009; Wang et al., 2019a; Ying et al., 2018; Yuan et al., 2020; Hu et al., 2020b; Qiu et al., 2020; Wang et al., 2020a; Jin et al., 2020; Zügner et al., 2018; Lai et al., 2020) have been proposed to reach state-of-the-art performances in various tasks. Typically, GAT (Velickovic et al., 2018) introduces the attention mechanism into the information aggregation process. SGC (Wu et al., 2019) simplifies the original Graph Convolutional Network (Kipf and Welling, 2017) by linearly propagating information and collapsing weights among graph layers. APPNP (Klicpera et al., 2019) extends the utilized neighborhood for node representation and achieves an adjustable neighborhood for classification.

Many existing GNNs are evaluated on graph benchmarks that are randomly split (Kipf and Welling, 2017; Xu et al., 2019), *i.e.*, the training and testing data share similar data distribution. However, in real-world applications, *e.g.*, recommender systems, training samples are observed and collected with selection bias, leading to inconsistencies between the training and testing distribution. Few works in the literature investigate such a real-world problem. Recently, GNM (Zhou et al., 2019) investigated a related problem named non-ignorable non-response, which indicates that the unlabeled nodes are missing not at random (MNAR). However, they only consider distribution shifts caused by node labels and neglect distribution shifts related to node attributes, and they solely discuss binary-class datasets in the experiments, which can be less practical. In this paper, we propose a framework that can alleviate the negative effects of shifts related to both labels and attributes, and obtains stable predictions on various graph benchmarks and real-world recommendation datasets. (Feng et al., 2020) propose to augment the graph data by random node dropping and drive the final representations obtained from different augmented views to be consistent. We differ this work by performing biased node selection rather than random drop, and by identifying the stable properties in the representation learning of each node rather than ensuring the consistency of final representations.

Many methods (Bickel et al., 2009; Dudík et al., 2005; Huang et al., 2006; Liu and Ziebart, 2014; Shimodaira, 2000) are proposed to enhance model stability and robustness against distribution shifts or selection bias. Traditional methods align the training data distribution and the testing data distribution by re-weighting the training samples. Kuang *et al.* (Kuang et al., 2018, 2020) define the stable prediction problem and set two quantitative goals. They propose sample re-weighting methods via variable decorrelation for isolating the effect of each predictor, thus recovering the causation between predictors and outcome variable. (Fan et al., 2022, 2021) follow (Kuang et al., 2018) to decorrelate different dimensions of the GNN output, and use linear models for prediction. These methods require prior knowledge of testing distribution, binary prediction, or linear models, which might not be satisfied in real-world graph applications. (He et al., 2020b) propose to build a static graph that captures general relational patterns. However, this work faces the challenge of being tightly coupled with the set generation problem and might not be transferred to a broader range of graph tasks. In this paper, instead of borrowing generic off-the-shelf stable learning tools where the assumptions might not be satisfied in the graph domain, we propose to capture stable properties in neighborhood aggregation of GNNs for stable prediction.

Our proposed method is motivated by the goal of learning invariant representations in the context of GNNs. Existing invariance learning methods have made significant contributions to the broader field. For example, IRM (Arjovsky et al., 2019) focuses on estimating invariant correlations across multiple training distributions. It aims to learn a data representation such that the optimal classifier, built upon that representation, performs consistently across different training distributions. ZIN proposed in (Arjovsky et al., 2019) tackles the challenge of learning invariant features without relying on environment partitioning. It introduces a framework to jointly learn the environment partition and invariant representations using additional auxiliary information. HRM (Liu et al., 2021) focuses on learning invariant relationships in the presence of unobserved heterogeneity among the data. While our work draws inspiration from invariance learning methods, it contributes specific techniques and considerations for learning invariant representations in the context of graph neural networks. We address the unique challenges posed by graph-structured data and provide novel methods that leverage the relational dependencies and structural invariances inherent in graphs.

3 Problem Formulation

Let $\mathcal{X}$ denote the space of observed features, $\mathcal{A}$ denote the space of adjacency matrix, and $\mathcal{Y}$ denote the outcome space. Following (Peters et al., 2016), we define a graph **environment** as the joint distribution P_{XAY} on $\mathcal{X} \times \mathcal{A} \times \mathcal{Y}$ and use $\mathcal{E}$ to denote the set of all environments. For each environment, we have a graph dataset $G^e = (\mathbf{X}^e, \mathbf{A}^e, Y^e)$, where $\mathbf{X}^e \in \mathcal{X}$ are node features, $\mathbf{A}^e \in \mathcal{A}$ is the adjacency matrix, and $Y^e \in \mathcal{Y}$ is the response variable (*e.g.*, node labels in the node classification problem). The joint distribution of features and outcomes on $(\mathbf{X}, \mathbf{A}, Y)$ can vary across environments, *i.e.*, $P^e_{XAY} \neq P^{e'}_{XAY}$ for $e, e' \in \mathcal{E}$, and $e \neq e'$. In this paper, we aim to achieve stable prediction across non-IID testing environments $\mathcal{E}^{te}$. Following (Kuang et al., 2018), we introduce two quantitative goals for stable prediction on

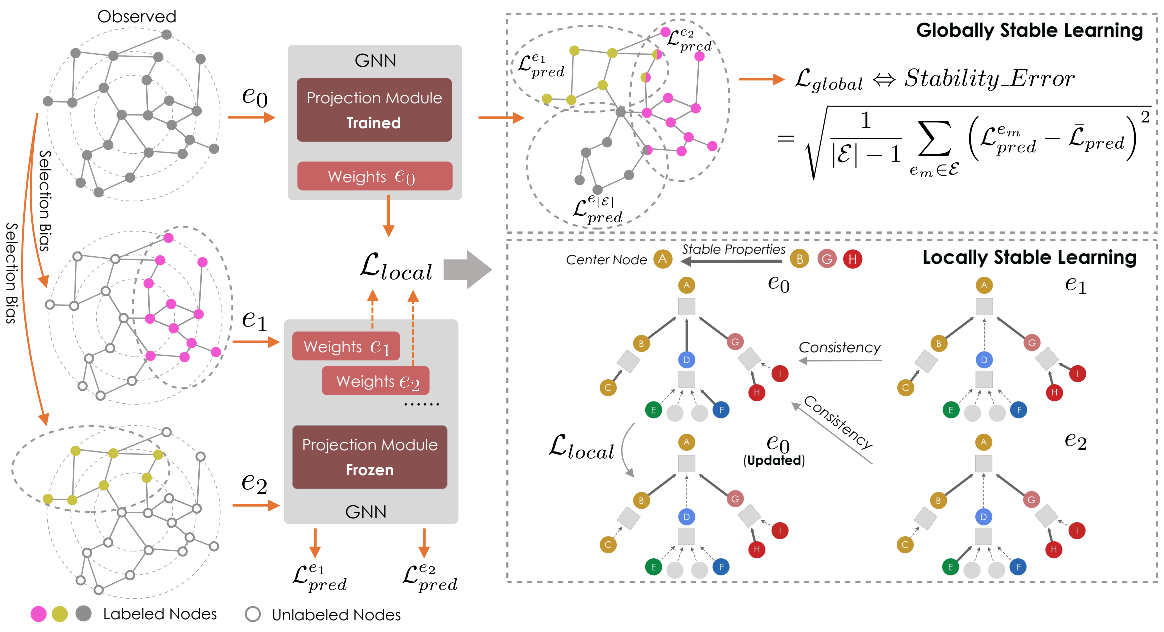


Figure 1: The overall schema of the proposed framework, which consists of two essential components, *i.e.*, the *locally stable learning* that captures properties that are stable across environments in the representation learning of each target node, and the *globally stable learning* that explicitly balances the training of different environments.

graphs as follows:

$$AVG\_Sco = \frac{1}{|\mathcal{E}^{te}|}\sum_{e \in \mathcal{E}^{te}} \mathrm{S}\left(G^e\right), \tag{1}$$

$$STB\_Err = \sqrt{\frac{1}{|\mathcal{E}|-1}\sum_{e \in \mathcal{E}^{te}} \left(\mathrm{S}\left(G^e\right) - AVG\_Sco\right)^2}, \tag{2}$$

where $|\mathcal{E}^{te}|$ denotes the number of non-IID testing environments, and $\mathrm{S}(G^e)$ refers to the predictive score reflecting the model performance on dataset G^e. As such, AVG_Sco indicates the average performance of the learned GNN in non-IID testing environments, and STB_Err indicates the performance variance of the learned GNN in non-IID testing environments. Based on these two metrics, we define the problem of stable prediction on graphs:

Problem 1 (Stable Prediction on Graphs) ***Given*** *one training environment e_0 with dataset $G^{e_0} = (\mathbf{X}^{e_0}, \mathbf{A}^{e_0}, Y^{e_0})$, the task is to learn a stable GNN that makes predictions with high AVG_Sco but low STB_Err across non-IID testing environments $\mathcal{E}^{te}$.*

4 Methods

4.1 A Retrospect of Modern GNNs

Here we briefly summarize typical GNNs and then give an illustration on why they suffer from distribution shifts. Modern GNNs follow a neighborhood aggregation schema by iteratively aggregating representations of neighbors to update the representation of the target node. One iteration can be generally formulated as:

$$\mathbf{x}_i^* = \text{AGGREGATE}\left(\{\mathbf{x}_j : j \in \mathcal{N}_i\}\right), \tag{3}$$

$$\mathbf{x}_i' = \text{COMBINE}\left(\mathbf{x}_i^*, \mathbf{x}_i\right), \tag{4}$$

where $\mathcal{N}_i$ denotes the indices of nodes in the neighborhood of node i. $\mathbf{x}_i^*$ denote the aggregated information of neighbor nodes for the target node i, and $\mathbf{x}_i'$ denotes the updated representation for node i. We denote the trainable parameters in GNN as θ. Predictions of GNNs are made based on the aggregated information from neighbors. In this light, we refer to neighbor nodes $\mathcal{N}_i$ as the **properties** of the target node i for prediction. Most GNNs are trained to maximize the correlation between properties and the labels in the given observational environment, *i.e.*, $G^{e_0} = (\mathbf{X}^{e_0}, \mathbf{A}^{e_0}, Y^{e_0})$. However, some correlations might be spurious (Pearl et al., 2016) and hardly generalize to non-IID testing environments. For example, in a paper citation network (*e.g.*, OGB-Arxiv), papers that cite LSTM (property) mostly have paper subject NLP (label) in the past training graph. Such correlation can hardly generalize to the current testing graph, where papers that cite LSTM have various subjects due to broadened impact of LSTM.

4.2 Analysis of Stable Prediction on Graphs

In this work, our goal is to make stable predictions against distribution shifts. According to (Kuang et al., 2018, 2020), the key to stable prediction lies in capturing stable properties for prediction, which can be summarized as the following hypothesis:

Hypothesis 1 *If we consider all stable properties (i.e., direct causes) of the node label (i.e., outcome), then the conditional distribution of the outcome given the stable properties will not change under interventions (e.g., biased selection).*

The essence of the above hypothesis lies in the invariance[2] of the correlation between stable properties and the label against *interventions*[3] and distribution shifts. To capture such invariance and achieve stable prediction on graphs, we set the goal as capturing stable properties in neighborhood aggregation of GNNs, and using GNNs to automatically learn the correlation between these properties and the node labels.

A fundamental question is how to identify stable properties in nodes' neighbors, or what kinds of properties are stable? Peters *et al.* (Peters et al., 2016) provide an intuitive

[2]Such invariance is closely linked to causality and has been discussed or empirically demonstrated to be effective in the literature (Haavelmo, 1944; Aldrich, 1989; Pearl et al., 2016).

[3]In particular, we use interventions to refer to the well-known do-interventions (Pearl et al., 2016), which correspond to fixing the intervened variable at different values, resulting in environments with distribution shifts. The intervened variables typically include the label and the covariates that affect the properties, causing label shifts and covariate shifts on distributions.

criterion that properties that are consistently important under interventions are stable ones. Inspired by this criterion, we regard nodes' neighbors that are consistently important across non-IID graph environments. In summary, the requirements for stable prediction on graphs can be decomposed as follows:

- *Non-IID Graph Environments.* In many cases, we have no access to multiple off-the-shelf non-IID graph environments, and have solely one observed training environment. We should intervene (label or covariate) variables and construct non-IID graph environments from the observed one.

- *Determination of Importance.* We need to measure the importance of each property for prediction *w.r.t.* each constructed graph environment. Ideally, an important property should contribute to not only the overall performance (high *Average_Score*) but also the stability against distribution shifts (low *Stability_Error*)

- *Stable Property Identification.* Based on the importance of properties across non-IID graph environments, we can identify those consistently important as stable properties.

- *Stable Prediction on Graphs.* Based on the identified stable properties, we can use GNNs to aggregate properties through neighborhood aggregation and make stable predictions.

4.3 Stable Learning on Graphs

To fulfill the above requirements in a unified end-to-end manner, we propose the stable learning on graphs framework. In the following, we dive into the technical details of several critical components of the framework, including the construction of non-IID graph environments, the property importance predictor, the locally stable regularizer for softly selecting stable properties, and the globally stable regularizer, which trains the importance predictor and the GNNs for pursuing low performance variance.

4.3.1 Constructing non-IID Graph Environments

Given the observed training graph environment $G^{e_0} = (\mathbf{X}^{e_0}, \mathbf{A}^{e_0}, Y^{e_0})$, we divide it into several non-IID graph environments $\{G^{e_k} = (\mathbf{X}^{e_k}, \mathbf{A}^{e_k}\ , Y^{e_k})\}_{k=1}^{|\mathcal{E}|}$ through biased selection *w.r.t.* some node attribute or node label. For example, given a recommendation user-item graph, we can construct one graph where the majority of user nodes are male and another graph where the majority of user nodes are female. Apparently, the graph data (such as connectivity patterns between user nodes and item nodes) of these two graphs are not IID due to the interest discrepancy between male and female users in recommender systems.

4.3.2 Property Importance Predictor

For each environment $e_k \in \{e_0, e_1, \ldots, e_{|\mathcal{E}|}\}$, we employ a neural network based importance predictor to assign weights to properties. Inspired by Graph Attention Network (Velickovic et al., 2018), the importance of neighbor node (property) x_j for the target node x_i can be modeled as follows:

$$\begin{aligned}\alpha_{ij}^{e_k} &= \varphi^{e_k}(\mathbf{x}_i, \mathbf{x}_j) &(5)\\ &= \frac{\exp\left(\text{ LeakyReLU }\left(\mathbf{a}^{e_k}\left[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j\right]\right)\right)}{\sum_{t\in\mathcal{N}_i}\exp\left(\text{ LeakyReLU }\left(\mathbf{a}^{e_k}\left[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_t\right]\right)\right)}, &(6)\end{aligned}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote the vectorial representation of nodes x_i and x_j. LeakyReLU denotes a nonlinear activation function (with negative input slope 0.2). α_{ij}^{e} is the weight that indicates the importance of one property x_j to the target node x_i learned in environment e. $\mathbf{a}^{e_k}$ denotes the parameter vector to determine the weight $\alpha_{ij}^{e_k}$ for graph environment e_k. $\mathcal{N}_i$ denotes the index set of node x_i's neighbors. The weights in graph attention indicate relative importance in the neighborhood, and it is also acceptable to use the following formulation, which has a sense of absolute importance:

$$\alpha_{ij}^{e_k} = \text{ sigmoid }\left(\mathbf{a}^{e_k}\left[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j\right]\right). \tag{7}$$

Note that each environment e_k has a different parameter vector $\mathbf{a}^{e_k}$, which helps find environment-specific important properties.

4.3.3 Locally Stable Regularizer

Based on the predicted importance weights, we aggregate the information of neighbor nodes for representing the target node, *i.e.*,

$$\mathbf{x}_i^{e_k} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{e_k}\mathbf{W}\mathbf{x}_j\right), \tag{8}$$

where $\mathbf{x}_i^{e_k}$ denotes the updated representation of the target node x_i in the e_k environment, $\mathbf{W}$ denotes the weight matrix of the linear transformation, σ denotes the nonlinear activation function. Predictions such node classification and item recommendations are made based on the node representations $\mathbf{x}_i^{e_k}$. In order to achieve stable prediction, we should identify those properties that are consistently important across environments. In this work, we adopt a soft selection mechanism, *i.e.*, down-weighting inconsistent properties via a locally stable regularizer. Technically, the locally stable regularizer tries to pull closer the weights across constructed non-IID environments $e_1, e_2, \ldots, e_{|\mathcal{E}|}$ and the weight predicted in the observed environment e_0, *i.e.*,

$$\mathcal{L}_{local} = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\sum_{k\in\{1,2,\ldots,|\mathcal{E}|\}} d(\alpha_{ij}^{e_0}, \alpha_{ij}^{e_k}), \tag{9}$$

where $\mathcal{V}$ denotes the index set of all nodes, and $d(\cdot)$ denotes a distance function which is set as the L2 distance in our experiment for its empirical effectiveness.

This objective is to drive the weights in the observed environment to be consistent with the majority of the weights in the non-IID environments. By doing so, we aim to mitigate the impact of inconsistent properties across different environments. Intuitively, properties that are found less important in some environments will be down-weighted in the observed environment, and properties that are consistently important will obtain large weights. By emphasizing the common patterns and down-weighting the inconsistent properties, we encourage the model to focus on the features that are more relevant and reliable for the majority of environments. The locally stable regularizer will solely optimize the weight predictor φ^{e_0} in the observed environment e_0, which will be used during inference. When there are multiple GNN layers equipped with the re-weighting module, the regularizer can be written as:

$$\mathcal{L}_{local} = \sum_{l=1}^{N_l} \mathcal{L}_{local}^{l}, \tag{10}$$

where N_l denotes the number of GNN layers equipped with the re-weighting module.

4.3.4 Globally Stable Regularizer

The locally stable regularizer works under an underlying assumption, *i.e.*, the weight predictors learn the desired importance. Originally, these weight predictors are optimized according to the task-specific objective, such as node classification loss. As such, the learned weights pursue the overall performance for all environments, which corresponds to the first goal of stable prediction, *i.e.*, improving *Average_Score* (*c.f.* Section 3). However, an important property should also contribute to the second goal, *i.e.*, reducing *Stability_Error*, to achieve stable prediction. In this regard, besides the original task-specific objective, we introduce a globally stable regularizer, which explicitly reduces the standard deviation of losses across environments as follows:

$$\bar{\mathcal{L}}_{pred} = \frac{1}{|\mathcal{E}|} \sum_{e_m \in \mathcal{E}} \mathcal{L}_{pred}^{e_m}, \tag{11}$$

$$\mathcal{L}_{global} = \sqrt{\frac{1}{|\mathcal{E}|-1} \sum_{e_m \in \mathcal{E}} \left(\mathcal{L}_{pred}^{e_m} - \bar{\mathcal{L}}_{pred}\right)^2}, \tag{12}$$

where $\mathcal{L}_{pred}^{e_m}$ denotes the task-specific loss for nodes belonging to environment e_m, *i.e.*,

$$\mathcal{L}_{pred}^{e_m} = \frac{1}{|\mathcal{Y}_L^{e_m}|} \sum_{s \in \mathcal{Y}_L^{e_m}} \mathcal{L}_{pred,s}^{e_m}, \tag{13}$$

where $\mathcal{Y}_L^{e_m}$ denotes the index set of nodes that belong to environment e_m. $\mathcal{L}_{pred,s}^{e_m}$ denotes the prediction loss for the s-th node. Note that loss is an indicator of model performance, and $\mathcal{L}_{global}$ corresponds to the *Stability_Error* defined in Equation (2). $\mathcal{L}_{global}$ affects not only the weight predictor but also the GNN backbone, to make explicitly GNN pursue stable prediction at the environment level. In summary, the training objective for environment e_k $(k = 0, \ldots, |\mathcal{E}|)$ can be written as follows:

$$\mathcal{L}^{e_k} = \mathcal{L}_{pred}^{e_k} + \lambda_0 \mathcal{L}_{local} + \lambda_1 \mathcal{L}_{global}. \tag{14}$$

where λ_0 and λ_1 are hyperparameters to control the strength. Note that $\mathcal{L}_{local}$ will solely optimize the weight predictor φ^{e_0} in the observed environment e_0 (*c.f.* Section 4.3.3), which will be used during inference.

4.3.5 Training

We give an illustration of how we train the entire framework. Given one observational environment e_0 for training with dataset $G^{e_0} = (\mathbf{X}^{e_0}, \mathbf{A}^{e_0}, Y^{e_0})$, we perform biased selection with one or more factors (*e.g.*, node label or semantic node attributes) on G^{e_0} at the beginning of training or each training epoch. After selection, we obtain constructed environments $\mathcal{E}$ with the corresponding graph datasets $\{G^{e_k} = (\mathbf{X}^{e_k}, \mathbf{A}^{e_k}, Y^{e_k})\}_{k=1,\dots,|\mathcal{E}|}$. We train one importance weight predictor φ^{e_k} for $k = 0, \dots, |\mathcal{E}|$. All the environments share the same GNN backbone, where the parameters θ are frozen in constructed environments, and trained in the observed environment e_0. We train the entire framework following an environment-by-environment procedure with objective $\mathcal{L}^{e_k}$ defined in Equation (14). The training procedure is summarized in Algorithm 1. During inference, the GNN backbone and the weight predictor φ^{e_0} in the observed environment will be used.

Algorithm 1: Stable Learning on Graphs

```
Input: Observed graph environment $G^{e_0} = (\mathbf{X}^{e_0}, \mathbf{A}^{e_0}, Y^{e_0})$
Output: Parameters of GNN backbone $\theta$ and weight predictor $\varphi^{e_0}$
Biased selection on $G^{e_0}$ and obtain $\{G^{e_k} = (\mathbf{X}^{e_k}, \mathbf{A}^{e_k}, Y^{e_k})\}_{k=1}^{|\mathcal{E}|}$. Initialize $\theta$ and $\{\varphi^{e_k}\}_{k=0}^{|\mathcal{E}|}$
while not converged do
    for $k$ = 1 to $|\mathcal{E}|$ do
        Optimize $\varphi^{e_k}$ over $\mathcal{L}_{pred}^{e_k}$                ▷ Eq. (13)
    end
    Optimize $\varphi^{e_0}$ and $\theta$ over $\mathcal{L}^{e_0}$                 ▷ Eq. (14)
    Optimize $\{\varphi^{e_k}\}_{k=1}^{|\mathcal{E}|}$ over $\mathcal{L}_{global}$              ▷ Eq. (12)
end
```

5 Experiments

To evaluate the prediction stability of models, we construct non-IID testing datasets with biased selection. We also perform biased selection on the training dataset to have multiple training environments such that the distribution shifts from training to testing can be various, leading to a comprehensive analysis. We consider distribution shifts caused by node labels and attributes, *i.e.*, label shifts and covariate shifts.

Following previous GNNs (Velickovic et al., 2018; Kipf and Welling, 2017) , we evaluate the proposed stable learning framework on public graph benchmarks in Section 5.4. Since these datasets seldom accompany meaningful attributes, we consider *label* shifts for evaluation. We conduct hyper-parameter analysis to obtain a better understanding of the designs. We further introduce a real-world industrial recommendation dataset in Section 5.5, where recommender systems are known to be full of sample selection biases (Chen et al., 2020) and distribution shifts. We consider *covariate* shifts for evaluation on this dataset. We also introduce real-world non-IID testing environments for evaluation in Section 5.5.

Table 1: (a) Statistics of datasets. (b) Detailed statistics of the recommendation dataset, which contains five consecutive days collected during a product promotion festival.

(a)

Dataset	**Citeseer**	**OGB-Arxiv**	**Recommendation**
Nodes	3,327	169,343	29,444
Edges	4,732	1,166,243	180,792

(b)

Users	**Items**	**Iterac1**	**Iterac2**	**Iterac3**	**Iterac4**	**Iterac5**
9,052	20,392	180,792	161,035	174,511	181,166	233,373

5.1 Datasets and Preprocess

- *OGB-Arxiv*[4]. We obtain the OGB-Arxiv dataset from the Open Graph Benchmark team (Hu et al., 2020a). We use the training/validation/testing split provided by them. The OGB datasets are proposed with real-world (non-IID) training/testing splits, which are suitable for stability evaluation. Besides the original split, we perform biased selection on the first half of the training set based on labels (*i.e.*, label shift) to construct the observational environment. We construct non-IID testing environments from the remaining half training set. For each environment, the selection probability of node i with label y can be $P(s_i = 1) = \tau$ if $y \geq 24$ and $1-\tau$ otherwise. τ denotes the bias ratio indicating the severity of sample selection bias. We use τ^{tr}, τ^{te} to denote the training and testing bias ratio, respectively. In the experiments, we vary the training bias ratio τ^{tr} among $\{0.9, 0.8, 0.7\}$ indicating heavy, medium, and light selection biases, following (Kuang et al., 2018).

- *Citeseer*[5]. We obtain the Citeseer dataset from the Deep Graph Library (Wang et al., 2019b). Since the original training set will be relatively small after biased selection, we randomly select 400 samples from the original validation set into the training set. The remaining samples are used for validation. In constructing non-IID training/testing environments, the selection probability of node i with label y can be $P(s_i = 1) = \tau$ if $y \geq 3$ and $1 - \tau$ otherwise. We also report the results under the original data split according to (Sen et al., 2008).

- *Recommendation Dataset.* We collected an industrial dataset from Mobile Taobao during the period of June 11th, 2020 to June 15th, 2020, when an annual product promotion festival is being celebrated. In such a period, the promotion strategies from online shop owners can be various and time-evolving. Therefore, inconsistencies between the users' clicks and satisfaction naturally exist, leading to distribution shifts from the collected data and the real-world testing environment. We select users and items that have interactions in all the five days. We mainly consider click interactions, which is a common setting in the deep candidate generation phase in recommendation. The statistics are listed in

[4] https://github.com/snap-stanford/ogb

[5] https://github.com/dmlc/dgl/tree/master/python/dgl/data

Table 1. We perform biased selection to construct non-IID testing environments *w.r.t.* the gender and age attributes of users. There are 8 age sections in the dataset, including 1-18, 19-25, 26-30, 31-35, 36-40, 41-50, 51-60, and >=61. We group users of 1-18 and 19-25 into a section and the remaining users into another section. This split results in approximately equally-sized sections. The user-item bipartite graph consists of user nodes and item nodes, which are connected according to interactions.

5.2 Baseline

We consider various baselines for comparison. We use their official implementations for most baselines (*c.f.* Appendix A.1.4).

- *Generic SOTA GNNs.* We consider several state-of-the-art (SOTA) GNNs as comparison methods, including GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), SGC (Wu et al., 2019), and APPNP (Bojchevski et al., 2020).

- *SOTA GNNs that deal with selection bias.* We further add two comparison methods that are designed for alleviating selection biases, *i.e.*, GNM (Zhou et al., 2019), and GAT-DVD (Fan et al., 2022). We do not test them on OGB-Arxiv since they are not easily adaptable for large-scale datasets. We consider training bias ratio $\tau^{tr} = 0.8$.

  We follow (Kipf and Welling, 2017; Velickovic et al., 2018) to construct two-layer GCN and GAT. All other methods (including ours) also contain two graph layers for a fair comparison. All methods are with hidden size 250 for OGB-Arxiv and 64 for Citeseer, and learning rate 0.002.

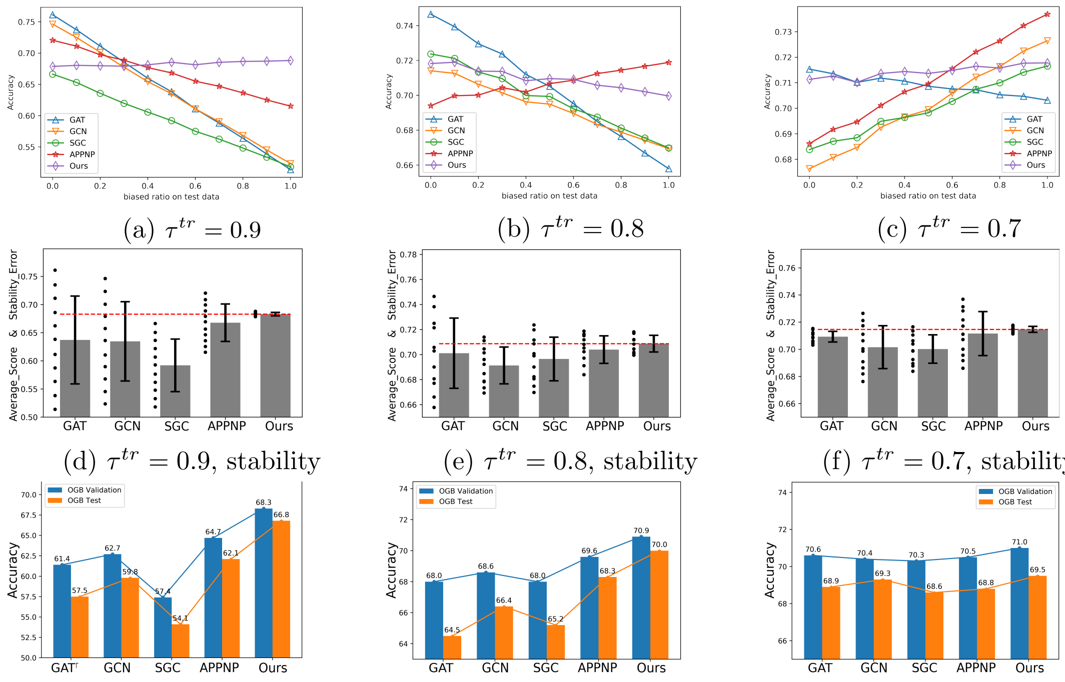
- *Graph Recommenders.* In the recommendation task, we consider two state-of-the-art graph-based recommendation models as baselines: 1) *NGCF*, which leverages GCN to represent users and items based on the user-item bipartite graph; and 2) *LightGCN*, which linearly propagates user/item embeddings on the user-item graph and thus simplifying the NGCF.

5.3 Evluation Protocol

- *OGB-Arxiv.* We use the evaluator provided by the official OGB Team (Hu et al., 2020a) and use node prediction accuracy as the performance score.

- *Citeseer.* We use node prediction accuracy as the score.

- *Recommendation.* Since we mainly focus on the matching phase of recommendation, we use widely used evaluation metrics (Wang et al., 2019c; He et al., 2020a), *i.e.*, Normalized Discounted Cumulative Gain (NDCG), and Recall. Recall considers how many groundtruth items are recommended in the Top-K list *w.r.t.* all groundtruth items. NDCG further considers the positions of recommended items in the Top-K list. Specifi-



(a) $\tau^{tr} = 0.9$ (b) $\tau^{tr} = 0.8$ (c) $\tau^{tr} = 0.7$

(d) $\tau^{tr} = 0.9$, stability (e) $\tau^{tr} = 0.8$, stability (f) $\tau^{tr} = 0.7$, stability

(g) $\tau^{tr} = 0.9$, official val/test (h) $\tau^{tr} = 0.8$, official val/test (i) $\tau^{tr} = 0.7$, official val/test

Figure 2: Testing results on OGB-Arxiv dataset by varying the training bias ratio τ^{tr} among $\{0.9, 0.8, 0.7\}$. Subfigures 2a - 2c show the metrics by varying the testing bias ratio. Subfigures 2d - 2f explictly show the stability of prediction by plotting the AVG_Sco and STB_Err. Subfigures 2g - 2i show the results on the original validation and testing datasets.

cally, NDCG can be formally written as:

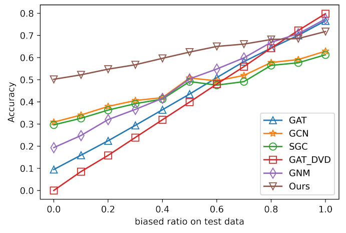
$$\text{DCG@N} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{r \in \hat{\mathcal{I}}_{u,N}} \frac{\mathbb{1}(r \in \mathcal{I}_u)}{\log_2 (i_r + 1)} \tag{15}$$

$$\text{NDCG@N} = \frac{\text{DCG@N}}{\text{IDCG@N}} \tag{16}$$

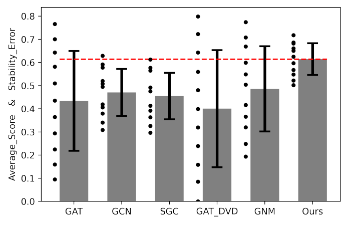
where $\mathcal{U}$ is the set of users, N is the number of recommended items, and $\hat{i}_{u,k}$ indicates the kth item recommended for user u. $\mathbb{1}$ denotes the indicator function. IDCG@N denotes the ideal discounted cumulative gain and is the maximum possible value of DCG@N.

5.4 Experiments on Graph Benchmarks

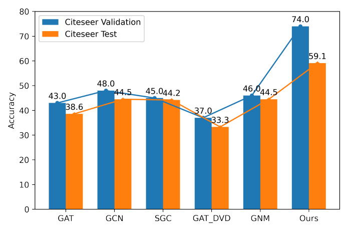
Stability comparison with SOTA GNNs. The testing results on the OGB-Arxiv dataset and Citeseer dataset are shown in Figure 2 and Figure 3, respectively. Specifically, Figure 2a - 2c plot the evaluation results *w.r.t.* non-IID testing environments with bias ratio $\tau^{te} \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$, and non-IID training environments with bias ratio $\tau^{tr} \in \{0.7, 0.8, 0.9\}$. Note that the height of each histogram refers to the AVG_Sco and the length



(a) $\tau^{tr} = 0.8$, varying τ^{te}



(b) $\tau^{tr} = 0.8$, stability



(c) $\tau^{tr} = 0.8$, official val/test

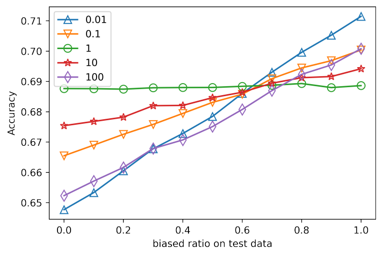
Figure 3: Testing results on the Citeseer dataset.

of each vertical line on the histograms refers to the STB_Err. According to the results, we have the following key observations:

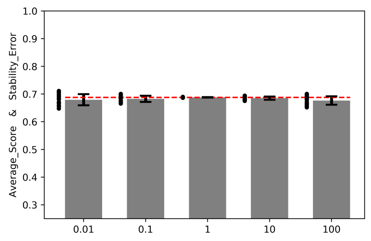
- In a nutshell, our framework achieves more stable results than state-of-the-art GNNs, including generic ones (GAT, GCN, SGC, and APPNP) and those designed for reducing selection biases (GAT-DVD and GNM). Under non-IID testing environments, we observe that most GNNs suffer from the distributional shifts and yield poorer performances when the training-testing distribution shift is more severe (*e.g.*, the right part of Figure 2a). Although our framework sacrifices some performance in testing environments with distribution closer to the training (*e.g.*, the left of Figure 2a), our framework obtains significantly higher AVG_Sco and lower STB_Err across non-IID environments, as indicated in Figure 2d - 2f, 3b, which are important indicators for stable prediction (*c.f.* Section 3) (Kuang et al., 2018).

- By varying the training bias ratio τ^{tr} from *light* to *heavy* (from Figure 2c to 2a), we can observe that most GNNs yield less stable results, *i.e.*, larger STB_Err (length of the vertical line) and significantly smaller AVG_Sco (height of histogram) . In contrast, our framework achieves consistently stable results with a minor AVG_Sco drop. We also notice that some baselines could occasionally achieve stable results. For example, as indicated in Figure 2f, GAT yields a smaller STB_Err than other baselines. However, when the training bias ratio is different (*e.g.*, $\tau^{tr} \in \{0.9, 0.8\}$), GAT yields significantly larger STB_Err than other baselines, indicating unstable predictions. We observe similar results for some other baselines like APPNP. In other words, these baselines cannot achieve consistently stable predictions across agnostic and heterogeneous distribution shifts.

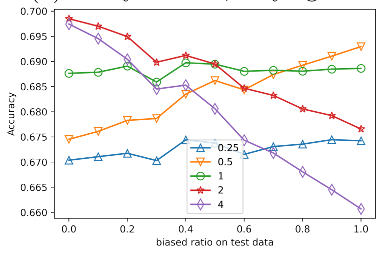
- Surprisingly, with the same hyper-parameters, GAT-DVD and GNM achieve more unstable results than other generic GNNs. As shown in Figure 3b, GAT-DVD and GNM yield similar or better AVG_Sco but significantly larger STB_Err compared to other state-of-the-art GNNs. GNM is designed for binary-class datasets and may perform poorly or need further improvements when extending to multi-classes datasets. These results probably indicate that decorrelation on the output of GNNs with linear predictors in GAT-DVD is not enough for stable prediction on graphs, requiring in-depth analysis of the internal behavior of GNNs.



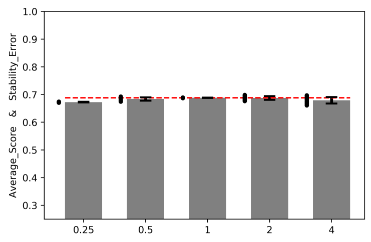
(a) Analysis of λ_0, varying τ^{te}



(b) Analysis of λ_0, statbility



(c) Analysis of λ_1, varying τ^{te}



(d) Analysis of λ_1, statbility

Figure 4: Hyper-parameter analysis on the OGB-Arxiv dataset with training bias ratio $\tau^{tr} = 0.8$.

Performance on the official validation/testing datasets. The official training/testing datasets in OGB-Arxiv are splits under real-world settings with distribution shifts (Hu et al., 2020a). We are interested in whether our framework could perform well under such settings. In this regard, we also report the performance on the official validation and testing datasets. As shown in Figure 2g-2i, our framework consistently outperforms the other methods under different non-IID training environments. The improvements over baselines are larger when the training selection bias is more severe. These results jointly indicate the rationality of our framework.

Analysis of Hyper-parameters. We are interested in how the coefficients λ_0, λ_1 of the proposed locally stable regularizer and globally stable regularizer in Equation (14) affect the prediction stability. We vary λ_0, λ_1 and obtain results on OGB-Arxiv dataset with training bias ratio $\tau^{tr} = 0.8$, as shown in Figure 4. When increasing the λ_0 and λ_1 from a lower rate, we observe stability improvement, *i.e.*, improved AVG_Sco and reduced STB_Err. This demonstrates the effectiveness of our framework in a sense of ablation study. We observe a slight performance drop when λ_0 and λ_1 further increase. We attribute the phenomenon to that the dominance of one regularizer might deteriorate the capability of the other regularizer as well as the task-specific objective. Overall, the prediction stability is insensitive to these hyperparameters.

5.5 Experiments on a Recommendation Dataset

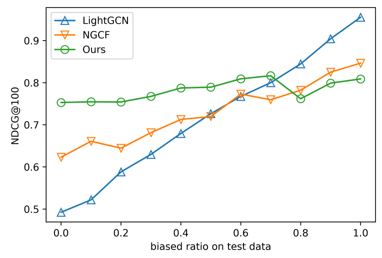
In this section, we evaluate the proposed stable learning framework on graph-based recommendation. We conduct experiments on a noisy industrial recommendation dataset collected from Mobile Taobao where the details can be found in Section 5.1.

Table 2: Results on recommendation dataset with real-world environments (each day as an individual testing environment). ***Large*** *AVG_Sco* and ***small*** *STB_Err* indicate better performance for stable prediction. The STB_Err values are with 10^{-1}.

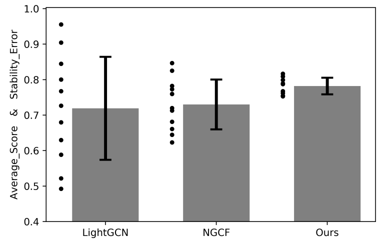
		Gender Bias					
Model	Metric	Day2	Day3	Day4	Day5	**AVG_Sco**	**STB_Err**
NGCF	Recall@20	0.1839	0.1805	0.1848	0.1755	0.1812	0.3646
	NDCG@20	0.2101	0.1918	0.1864	0.1947	0.1957	0.8795
LightGCN	Recall@20	0.1539	0.1203	0.1542	0.1221	0.1376	1.6433
	NDCG@20	0.1481	0.1273	0.1495	0.1431	0.1420	0.8817
Ours	Recall@20	0.1935	0.1859	0.1849	0.1867	**0.1877**	**0.3386**
	NDCG@20	0.2112	0.2177	0.2103	0.2151	**0.2136**	**0.3013**
		Age Bias					
Model	Metric	Day2	Day3	Day4	Day5	**AVG_Sco**	**STB_Err**
NGCF	Recall@20	0.4012	0.3395	0.3174	0.2925	0.3377	4.0295
	NDCG@20	0.4335	0.3755	0.3768	0.3893	0.3938	2.3578
LightGCN	Recall@20	0.4467	0.4029	0.3582	0.3121	**0.3800**	5.0161
	NDCG@20	0.4486	0.4267	0.4273	0.4558	**0.4396**	1.2878
Ours	Recall@20	0.3627	0.4086	0.3561	0.3218	0.3623	**3.0909**
	NDCG@20	0.4197	0.4207	0.4096	0.4248	0.4187	**0.5595**

Analysis of Stable Prediction. We group the data samples collected from the five consecutive days as a holistic dataset. We use 80% interactions of each user for training and the remaining 20% interactions for testing. We consider two kinds of shifts between training distribution and testing distribution caused by user gender and user age, respectively. For each attribute, we set training bias rate $\tau^{tr} = 0.6$ and test the model on testing environments with bias rate τ^{te} varying among $\{0.0, 0.1, 0.2, \ldots, 1.0\}$. Detailed preprocessing can be found in Section 5.1. According to Figure 5, we can find that the proposed method yields significant improvement on AVG_Sco (height of the histogram), and simultaneously reduces the STB_Err (length of the vertical line) over state-of-the-art graph-based recommendation models across different testing environments. These two metrics jointly demonstrate that the proposed method is stable against distribution shifts between training and testing. We attribute these merits to capturing stable properties for representing users and items on the interaction graph. Note that the interactions between users and items (such as clicks) in recommender systems are noisy (O'Mahony et al., 2006) due to external distractions such as product promotion. Such a problem becomes more severe since the dataset is collected during a product promotion festival. It is essential to capture stable interactions such that we can accurately understand users and items for stable recommendation.

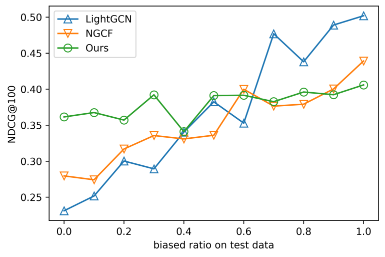
Evaluation on Real-world Testing Environments. Due to the rapid change of product promotion strategies, there are distribution shifts among the data samples of different



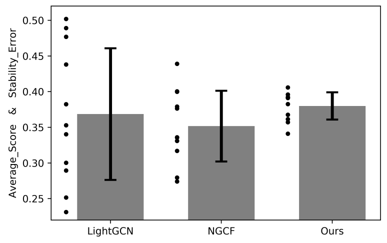
(a) Age, varying τ^{te}



(b) Age, stability



(c) Gender, varying τ^{te}



(d) Gender, stability

Figure 5: Results on real-world recommendation dataset with distribution shifts caused by node attributes.

days, which constitute real-world non-IID testing environments. To reveal the model performance under such settings, we train a model on the data samples collected in Day1, and report the recommendation performance in Day2-Day5, respectively. In addition, we report the AVG_Sco and STB_Err, as defined in Section 3, of each model to demonstrate the prediction stability across non-IID environments. According to the results shown in Table 2, we have several observations.

- In a nutshell, the proposed method yields performance improvement over baselines in most testing environments. These results basically indicate that the proposed method suffers less from the distribution shifts, and can generalize to non-IID environments.
- As for prediction stability, the proposed method achieves the least STB_Err in all cases, and competitive AVG_Sco in most cases. Low STB_Err and high AVG_Sco mean that users receive consistently high-quality recommendation services under external distractions (*e.g.*, heterogeneous product promotion strategies) that cause distribution shifts. Undoubtedly, stable service quality is an essential merit for real-world industrial platforms.
- We take a step further to investigate the performance change by the degree of distribution shifts. Intuitively, testing environment Day5 is more likely to have a larger distribution shift from the training environment Day1 than Day2 due to potential correlation on product promotion strategies of adjacent days. We observe that the performance of all models exhibits a decreasing trend from Day2 to Day5 in many cases, which is intuitive

due to the increasingly larger distribution shifts. Interestingly, we find that performance improvement of the proposed method over two baselines becomes larger from Day2 to Day5 in many cases. For example, *w.r.t.* gender bias, the proposed method yields +0.0396 and +0.0646 Recall@20 improvement over LightGCN on Day2 and Day5, respectively. These results again indicate better stability of our method against distribution shifts over two representative graph recommendation models.

6 Conclusion and Future Work

In this paper, we study how to achieve stable prediction on graphs against agnostic distribution shifts. We argue that capturing stable properties in the neighborhood aggregation of GNNs could improve stable prediction. To achieve this target, we propose a stable learning framework for GNNs, which identifies stable properties with constructed non-IID environments and the globally stable regularizer, and down-weights unstable properties in prediction with the locally stable regularizer. We conduct extensive experiments on public graph benchmarks as well as a noisy industrial recommendation dataset. We show that our framework could achieve more stable prediction *w.r.t.* label shifts and covariate shifts, and *w.r.t.* real-world testing environments. We believe that the insights of the proposed framework are inspirational to future developments of stable learning techniques on graphs. We plan to further investigate whether causal discovery/inference techniques could help to disentangle the stable/unstable properties on graphs for stable prediction.

Acknowledgments and Disclosure of Funding

This work was supported in part by National Natural Science Foundation of China (62006207, U20A20387, 62037001), Key R & D Projects of the Ministry of Science and Technology (2020YFC0832500), Young Elite Scientists Sponsorship Program by CAST (2021QNRC001), Zhejiang Province Natural Science Foundation (LR19F020006, LQ21F020020), Project by Shanghai AI Laboratory (P22KS00111), Program of Zhejiang Province Science and Technology (2022C01044), the StarryNight Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010).

Appendix A. Appendix

Table 3: Hyper-parameters

Hyper-parameter	OGB-Arxiv	Citeseer	Rec.
Batch size	Full	Full	512
Training epochs	1,000	200	120
Initial learning rate	0.002	0.005	0.0001
Learning rate decay	Linear	None	None
Weight decay	None	5e-4	None
Adam ϵ	1e-8	1e-8	1e-8
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999
Number of layers	2	2	3
Hidden units per layer	250	8	64
Dropout rate	0.75	0.6	None
Input-drop rate	0.1	0.1	0.1
Edge-drop rate	0.1	None	None

A.1 Experiment Details

A.1.1 Hardware Configuration

The experiments are conducted on Linux servers equipped with an Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz, 330GB RAM and 8 NVIDIA Tesla V100-SXM2-16GB GPUs.

A.1.2 Software Configuration

Our framework is implemented in PyTorch (Paszke et al., 2019) with version 1.7.0, DGL (Wang et al., 2019b) with version 0.5.2, CUDA version 10.2, and Python 3.6.12. Code and datasets will be made publicly available.

A.1.3 Detailed Hyper-parameters

The detailed hyper-parameters are listed in Table 3.

A.1.4 Implementation of Baselines

- *Generic SOTA GNNs.* We consider several state-of-the-art (SOTA) GNNs as comparison methods, including GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), SGC (Wu et al., 2019), and APPNP (Bojchevski et al., 2020).
  - *GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018).* We download the source code provided by the DGL Team (Wang et al., 2019b). These implementations are tuned by the DGL Team especially for the OGB-Arxiv dataset. We set the

hyper-parameters the same as described in Table 3. Other hyper-parameters for each particular model are set as default in the source code[6].

  - *SGC (Wu et al., 2019).* We implement SGC by directly replacing the graph convolution operator in the above GCN implementation with the SGConv provided by the DGL Team (Wang et al., 2019b). All input parameters to construct the SGConv component stay the same as the GCNConv component.

  - *APPNP (Bojchevski et al., 2020).* We implement APPNP by adding one more APPNP layer onto the output layer of GCN model. We set hyperparameters: $K = 5$, teleport probability $\alpha = 0.1$ with edge dropout rate 0.1.

- *SOTA GNNs that deal with selection bias.*

  - *GAT_DVD (Zhou et al., 2019).* We download the authors' official source code[7] and change the hyper-parameter settings according to Table 3.

  - *GNM (Fan et al., 2022).* We download the authors' official source code[8]. Hyper-parameters are changed according to Table 3 and other model-specific hyper-parameters stay the same as default. This source code is designed for binary classifications. We make necessary modifications such as changing the number of classes to the corresponding number of each dataset.

- *Graph Recommenders.*

  - *NGCF.* NGCF leverages GCN to represent users and items based on the user-item bipartite graph. We use a Pytorch-based implementation[9] and use default hyper-parameters provided by the authors.

  - *LightGCN.* LightGCN linearly propagates user/item embeddings on the user-item graph and thus simplifying the NGCF. We use a Pytorch-based implementation[10] and use default hyper-parameters provided by the authors.

  - *Stable Graph Recommender.* We build the proposed stable graph recommender based on NGCF. The implementation can be found in Appendix A.2.

[6]`https://bit.ly/35xD4DC`
[7]`https://openreview.net/forum?id=xboZWqM_ELA.`
[8]`https://github.com/mlzxzhou/keras-gnm`
[9]`https://github.com/huangtinglin/NGCF-PyTorch`
[10]`https://github.com/gusye1234/LightGCN-PyTorch`

A.2 Stable Graph Recommender

NGCF largely follows the standard GCN model. The proposed stable graph recommender propagates embeddings on the user-item bipartite graph as follows:

$$\alpha_{ui}^{e} = \text{ sigmoid } \left(\mathbf{a}^{e}\left[\mathbf{e}_u^0 \| \mathbf{e}_i^0\right]\right) \tag{17}$$

$$\mathbf{e}_u^{(k+1)} = \sigma(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i\in\mathcal{N}_u} \frac{\alpha_{ui}^{e}}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_i^{(k)} \tag{18}$$

$$+ \mathbf{W}_2(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}))) \tag{19}$$

$$\mathbf{e}_i^{(k+1)} = \sigma(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u\in\mathcal{N}_i} \frac{\alpha_{ui}^{e}}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_u^{(k)} \tag{20}$$

$$+ \mathbf{W}_2(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)}))) \tag{21}$$

where $\mathbf{e}_u^{(k+1)}$ and $\mathbf{e}_i^{(k+1)}$ denote the updated user and item embedding after k layers propagation. σ denotes a nonlinear activation function and is LeakyReLU as NGCF. $\mathcal{N}_u$ denotes the set of interacted items for user u and $\mathcal{N}_i$ denotes the set of interacted users for item i. $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable transformation matrices. α_{ui}^{e} denotes the importance of interaction $< u, i >$ for user u and item i. Note that interactions that are consistently important across environments $\mathcal{E}$ are stable properties, as illustrated in Section 4.3.3. We keep a global α_{ui}^{e} for layers due to its efficiency and do not compute weights per layer. Note that deep candidate generation models, which recall Top K items from a billion-scale item gallery are largely sensitive to model efficiency. For example, one of the contributions of LightGCN is to remove the nonlinearity σ in NGCF and thus improving efficiency. We train the stable graph recommender as illustrated in Section 4.3.5.

References

John Aldrich. Autonomy. *Oxford Economic Papers*, 1989.

Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *CoRR*, abs/1907.02893, 2019.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *J. Mach. Learn. Res.*, 2009.

Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *CoRR*, 2020.

Miroslav Dudík, Robert E. Schapire, and Steven J. Phillips. Correcting sample selection bias in maximum entropy density estimation. In *Advances in Neural Information Processing Systems 18*, 2005.

Shaohua Fan, Xiao Wang, Chuan Shi, Peng Cui, and Bai Wang. Generalizing graph neural networks on out-of-distribution graphs. *CoRR*, abs/2111.10657, 2021.

Shaohua Fan, Xiao Wang, Chuan Shi, Kun Kuang, Nian Liu, and Bai Wang. Debiased graph neural networks with agnostic label selection bias. *CoRR*, abs/2201.07708, 2022.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *Advances in Neural Information Processing Systems 33.*, 2020.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017.*, 2017.

Trygve Haavelmo. The probability approach in econometrics. *Econometrica: Journal of the Econometric Society*, 1944.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, 2020a.

Yue He, Peng Cui, Jianxin Ma, Hao Zou, Xiaowei Wang, Hongxia Yang, and Philip S. Yu. Learning stable graphs from multiple environments with selection bias. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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. In *NeurIPS*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020b.

Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems 19*, 2006.

Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. Redundancy-free computation for graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Kun Kuang, Peng Cui, Susan Athey, Ruoxuan Xiong, and Bo Li. Stable prediction across unknown environments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, 2018.

Kun Kuang, Ruoxuan Xiong, Peng Cui, Susan Athey, and Bo Li. Stable prediction with model misspecification and agnostic distribution shift. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 2020.

Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. Policy-gnn: Aggregation optimization for graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Anqi Liu and Brian D. Ziebart. Robust classification under sample selection bias. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, 2014.

Jiashuo Liu, Zheyuan Hu, Peng Cui, Bo Li, and Zheyan Shen. Heterogeneous risk minimization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6804–6814. PMLR, 2021.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Sharon L Lohr. *Sampling: design and analysis.* Nelson Education, 2009.

Michael P. O'Mahony, Neil J. Hurley, and Guenole C. M. Silvestre. Detecting noise in recommender system databases. In *Proceedings of the 11th International Conference on Intelligent User Interfaces, IUI 2006*, 2006.

Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. Estimating node importance in knowledge graphs using graph neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019.*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer.* John Wiley & Sons, 2016.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 2016.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. Graph neural networks for friend ranking in large-scale social platforms. In *WWW '21: The Web Conference 2021.*, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 2008.

Zheyan Shen, Peng Cui, Tong Zhang, and Kun Kuang. Stable learning via sample reweighting. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 5692–5699. AAAI Press, 2020.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 2000.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Daheng Wang, Meng Jiang, Munira Syed, Oliver Conway, Vishal Juneja, Sriram Subramanian, and Nitesh V. Chawla. Calendar graph neural networks for modeling time structures in spatiotemporal user behaviors. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020a.

Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*, 2019a.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, and et al. Deep graph library: Towards efficient and scalable deep learning on graphs. *CoRR*, 2019b.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019.*, 2019c.

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. Am-gcn: Adaptive multi-channel graph convolutional networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020b.

Felix Wu, Jr. Souza, Amauri H., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 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, ICML 2018.*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, 2018.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*, 2019.

Fan Zhou, Tengfei Li, Haibo Zhou, Hongtu Zhu, and Ye Jieping. Graph-based semi-supervised learning with non-ignorable non-response. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, 2018.