FLOOD: A Flexible Invariant Learning Framework for Out-of-Distribution Generalization on Graphs

Yang Liu[†]
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
liuyang520ict@gmail.com

Xiang Ao[*†]
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
aoxiang@ict.ac.cn

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

Yunshan Ma
National University of Singapore
yunshan.ma@u.nus.edu

Kuan Li[†]
Institute of Computing Technology, Chinese Academy of Sciences
likuan_buaa@163.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

Qing He[*†]
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
heqing@ict.ac.cn

ABSTRACT

Graph Neural Networks (GNNs) have achieved remarkable success in various domains but most of them are developed under the in-distribution assumption. Under out-of-distribution (OOD) settings, they suffer from the distribution shift between the training set and the test set and may not generalize well to the test distribution. Several methods have tried the invariance principle to improve the generalization of GNNs in OOD settings. However, in previous solutions, the graph encoder is immutable after the invariant learning and cannot be adapted to the target distribution flexibly. Confronting the distribution shift, a flexible encoder with refinement to the target distribution can generalize better on the test set than the stable invariant encoder. To remedy these weaknesses, we propose a Flexible invariant Learning framework for Out-Of-Distribution generalization on graphs (FLOOD), which comprises two key components, invariant learning and bootstrapped learning. The invariant learning component constructs multiple environments from graph data augmentation and learns invariant representation under risk extrapolation. Besides, the bootstrapped learning component is devised to be trained in a self-supervised way with a shared graph encoder with the invariant learning part. During the test phase, the shared encoder is flexible to be refined with the bootstrapped learning on the test set. Extensive experiments are conducted for both transductive and inductive node classification tasks. The results demonstrate that FLOOD consistently outperforms other graph OOD generalization methods and effectively improves the generalization ability.

CCS CONCEPTS

• **Mathematics of computing** → *Graph algorithms*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

graph neural networks, out-of-distribution, invariant learning

ACM Reference Format:
Yang Liu, Xiang Ao, Fuli Feng, Yunshan Ma, Kuan Li, Tat-Seng Chua, and Qing He. 2023. FLOOD: A Flexible Invariant Learning Framework for Out-of-Distribution Generalization on Graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599355

1 INTRODUCTION

Graph Neural Network (GNN) has become a promising solution for various graph-based learning tasks, such as social recommendation [7, 34, 37], drug discovery [11, 17, 29], fraud detection [10, 18, 19, 32], and adversarial robustness [15, 16]. However, most GNNs have been developed under the in-distribution(ID) assumption and may not perform well in out-of-distribution(OOD) settings. The OOD learning deals with scenarios when training and testing data follow different distributions, which would happen where there are multiple graphs from different domains or the graph structure evolves with time.

[*]Corresponding author.
[†]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS). Xiang Ao is also at Institute of Intelligent Computing Technology, Suzhou, China.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599355

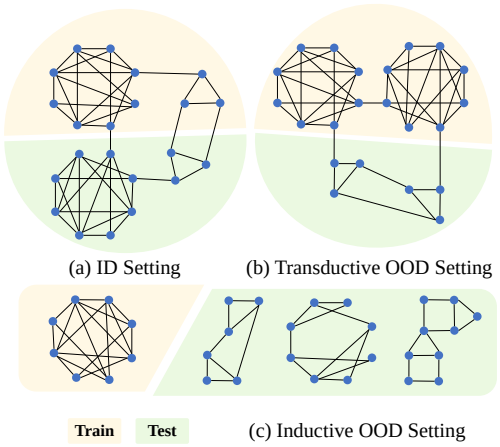


Figure 1: Demonstration of transductive and inductive OOD settings on graphs (Shift domain: node degree).

Focusing on the node classification task, the out-of-distribution settings on graphs can be categorized into transductive OOD settings and inductive OOD settings with regard to the availability of the test samples during the training phase. Transductive OOD settings have access to both the training and testing samples during the training phase, and a distribution shift would happen when the train-test split is correlated to a spurious domain. For example, for paper topic classification in a citation graph, the node degree is a spurious feature. If we deliberately choose nodes with larger degrees as training set and test the model on those nodes with smaller degrees, as shown in Figure 1(b), the model would not generalize well to the test set and suffer from the distribution shift. In contrast, a random split as in Figure 1(a) would not result in such a distribution shift. A typical example of transductive OOD datasets is GOOD [6] and it includes OOD splits for existing graph benchmarks. Inductive OOD settings do not allow access to testing samples during the training phase, thus the shift domain is unknown. A typical example of inductive OOD settings is the multi-graph generalization [36] as Figure 1(c) shows. When we train a GNN on the training graph, we need to test the model on multiple graphs from different sources. However, regardless of whether the OOD setting is transductive or inductive, test samples are always available during the test phase.

Several efforts have been conducted to improve the generalization of graph neural networks in out-of-distribution settings, employing various strategies such as pre-training, data augmentation, invariant learning, etc. Pre-training has rendered effective in many language and vision domains [2, 23]. Graph pre-training [8, 9], following the similar idea, designs different kinds of self-supervised tasks to capture the structural and semantic properties of the graph. The pre-trained GNN is expected to transfer the knowledge to downstream tasks with a few labels. One drawback of graph pre-training is the need for a large training corpus and a few labels of the targeted tasks. However, for out-of-distribution generalization, it is hard to get the labels of OOD samples. Graph data augmentation is mainly designed to address the issues of data noise and scarcity. It might also help OOD generalization as it can enrich the training distribution. Hence, graph augmentation methods are usually adopted as baseline for comparison in current graph OOD solutions. Invariant learning [1, 14, 36] aims to capture the invariant graph patterns across different training environments while disregarding the variant spurious correlation. Its generalization ability relies on the diversity of the training environments. However, current solutions based on invariant learning anticipate that a fixed graph encoder can output representations invariant to various OOD scenarios, and do not consider what the test distribution is like, which limits the generalization ability of the model.

Although several attempts [14, 36] have applied invariant learning on graphs to handle distribution shifts, they still suffer from the following drawbacks. The first is that it is hard to construct and optimize various training environments on graphs for invariant learning. EERM [36] designs multiple context generators to construct different virtual environments, but this incurs a high training cost, and the output of the generators cannot be guaranteed. GIL [14] infers the latent environment by clustering algorithms, which cannot enrich the training distribution. The second is that existing methods expect one model to generalize to various distributions and thus are not flexible. The model trained under the invariance principle is able to extrapolate to distributions around the training environments. However, the test distribution can still be out of the scope of extrapolation and the model needs the flexibility to be refined according to the target distribution.

To remedy the above weaknesses, we propose a Flexible invariant Learning framework for Out-Of-Distribution generalization on graphs (FLOOD). For the first weakness, we regard each augmented view of the original graph as a kind of training environment. The benefits are two-fold. On one hand, data augmentations are always computationally efficient and do not require many efforts to be trained. On the other hand, some data augmentations can improve the OOD generalization of GNN models, and we can benefit from the augmentation distribution by incorporating it into the invariant learning framework. For the second one, a bootstrapped learning part is devised to be trained in a self-supervised way with a shared graph encoder with the invariant learning part. During the test phase, the shared encoder is flexible to be refined with the bootstrapped learning on the test set. In this way, the representation output by the shared encoder can generalize well to the test distribution and mitigate the impact of the distribution shift.

Our contributions can be listed as follows.

- We investigate the graph out-of-distribution generalization problem and explore how to adapt the invariant model to the target distribution flexibly.
- We design a flexible invariant learning framework with a bootstrapped learning component that can refine the encoder during the test phase.
- Extensive experiments are conducted on graph OOD benchmark for both the transductive and inductive tasks to verify the effectiveness of the proposed framework.

2 PRELIMINARIES

In this work, let $\mathcal{X}$ be the input space, $\mathcal{Y}$ be the target label space, and $\mathcal{E}^{\text{obs}}$ be the set of training environments. Let $(x, y, e) \sim P_{\text{obs}}(x, y, e)$ be observational data, with $x \in \mathcal{X}$, $y \in \mathcal{Y}$, $e \in \mathcal{E}^{\text{obs}}$.

2.1 Distribution Shift

Definition 2.1 (Covariate Shift). In covariate shift, the input distributions have been shifted between the training and testing data, while the conditional distribution keeps the same. Formally, $P_{\text{train}}(x) \neq P_{\text{test}}(x)$ and $P_{\text{train}}(y|x) = P_{\text{test}}(y|x)$.

Definition 2.2 (Concept Shift). In concept shift, the conditional distributions have been shifted between the training and testing data, while the input distribution keeps the same. Formally, $P_{\text{train}}(x) = P_{\text{test}}(x)$ and $P_{\text{train}}(y|x) \neq P_{\text{test}}(y|x)$.

2.2 Invariant Learning

Out-of-distribution generalization refers to the ability to achieve low error rates on unseen test distributions. Invariant learning approaches are proposed to reveal invariant relationships between the inputs and targets across domains. The empirical risk minimization (ERM) solution is found by minimizing the global risk, expressed as the expected loss over the observational distribution.

$$\mathcal{R}_{\text{ERM}}(f_\psi) = \mathbb{E}_{P_{\text{obs}}(x,y,e)}\left[\ell(f_\psi(x), y)\right] \tag{1}$$

Invariant Risk Minimization (IRM) [1] includes a regularized objective enforcing simultaneous optimality of the same classifier $\omega \circ f$ in all environments. $\mathcal{R}_e$ is short for $\mathcal{R}_{\text{ERM}}$ in environment e.

$$\mathcal{R}_{\text{IRM}}(f_\psi) = \sum_{e\in\mathcal{E}^{\text{obs}}} \mathcal{R}_e(f_\psi) + \lambda \left\|\nabla_\omega \mathcal{R}_e(\omega \circ f_\psi)\right\| \tag{2}$$

Risk Extrapolation (REx) [13] is a form of robust optimization over a perturbation set of extrapolated domains as shown in Eq. (3). It shows that reducing differences in risk across training domains can reduce a model's sensitivity to distribution shifts.

$$\mathcal{R}_{\text{REx}}(f_\psi) = \max_{\substack{\sum_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e\in\mathcal{E}^{\text{obs}}} \lambda_e \mathcal{R}_e(f_\psi) \tag{3}$$

2.3 Problem Statement

We denote a graph as $\mathcal{G} = (\mathcal{V}, \mathbf{X}, \mathbf{A})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes, $\mathbf{X} \in \mathbb{R}^{N\times D}$ denotes node features, and $\mathbf{A} \in \{0,1\}^{N\times N}$ is an adjacency matrix representing the connections between nodes. Note that N and D represent the number of nodes and features, respectively. Besides, we denote the labels of the node as $\mathbf{Y} \in \{0,1\}^{N\times C}$ where C is the number of classes.

We focus on the out-of-distribution node classification, where the target is to learn a C-way classification function from the training graph $\mathcal{G}_{\text{train}}$ to predict the unlabeled nodes in the test graph $\mathcal{G}_{\text{test}}$. $\mathcal{G}_{\text{train}}$ and $\mathcal{G}_{\text{test}}$ follow different distributions (either covariate shift or concept shift). For transductive settings, $\mathcal{G}_{\text{train}}$ and $\mathcal{G}_{\text{test}}$ are on the same graph but denote different node sets. For inductive settings, $\mathcal{G}_{\text{train}}$ and $\mathcal{G}_{\text{test}}$ are different graphs. Formally,

$$f_\psi : (\mathbf{X}, \mathbf{A}) \to \mathbf{Y}, \tag{4}$$

where ψ denotes the parameters of the function. The parameters are typically learned over a set of labeled nodes by minimizing a classification loss such as cross-entropy. Moreover, we aim to utilize $\mathcal{G}_{\text{test}}$ to finetune ψ for better generalization during the test time.

3 METHODOLOGY

We start by motivating our method before explaining its details in this section. Many graph out-of-distribution generalization methods are built on the invariance principle. They explore various training environments to find the invariant relationships between features and labels while disregarding the variant spurious correlations. Many efforts have been devoted to constructing the environments to explore the possible distribution. However, such approaches neglect what the test distribution is like and cannot be adapted to different test distributions. To mitigate this issue, we expect to build a flexible invariant learning framework with a bootstrapped representation learning process, which can exploit the target distribution in a self-supervised manner. Consequently, the parameters can be finetuned by the self-supervised task under the test-time training [20, 31] mechanism, without relying on any label of the test set.

3.1 Overview

We illustrate the pipeline of the proposed framework on an example graph in Figure 2, which consists of two modules: invariant representation learning and bootstrapped representation learning.

In the invariant learning part, we adopt Variance Risk Extrapolation [13] to train the GNN model for OOD generalization. The training environments are constructed by data augmentation on graphs. In the bootstrapped learning part, the online encoder is jointly trained with the shared encoder and the classifier in the training phase. The shared encoder is separately updated in the test phase to obtain a better representation on the test set.

3.2 Invariant Representation Learning

Firstly, we construct multiple training environments from the original training graph $\mathcal{G}_{\text{train}} = (\mathbf{X}, \mathbf{A})$. We perform two typical graph augmentations, namely node feature masking [41] and DropEdge [24], denoted by $\eta(\cdot)$.

$$\eta_e(\mathbf{X}, \mathbf{A}) = \left(\widetilde{\mathbf{X}}_e, \widetilde{\mathbf{A}}_e\right), \qquad e = 1, \ldots, M \tag{5}$$

Next, we train a GNN encoder $f_\omega()$ to extract features from the graphs under different training environments. $f_\omega : (\mathbf{X}, \mathbf{A}) \to \mathbb{R}^d$ is a L-layer graph neural networks and outputs d-dimension representation for each node. In layer l ($l = 1, \ldots, L$), the representation for node i under environment e is defined by Eq. (6), where $\mathcal{N}_e(i)$ indicates the neighbor set of node i decided by $\widetilde{\mathbf{A}}_e$, and $\mathbf{h}_{e,i}^{(0)}$ is defined by $\widetilde{\mathbf{X}}_e$. The encoder parameterized by ω is shared by all the training environments.

$$\mathbf{h}_{e,i}^{(l)} = \text{AGGREGATE}\left(\mathbf{h}_{e,i}^{(l-1)}, \left\{\mathbf{h}_{e,j}^{(l-1)} \mid j \in \mathcal{N}_e(i)\right\} | \omega^{(l)}\right) \tag{6}$$

Following that, a GNN-based classifier $f_\psi : (\mathbb{R}^d, \mathbf{A}) \to \mathbf{Y}$ is trained to get the final prediction. The GNN parameterized by (ω, ψ) is trained by minimizing the cross-entropy loss defined by Eq. (7).

$$\mathcal{R}_e(\omega, \psi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} \mathbf{Y}_{ij} \log\left[f_\psi\left(f_\omega\left(\widetilde{\mathbf{X}}_e, \widetilde{\mathbf{A}}_e\right)\right)\right]_{ij} \tag{7}$$

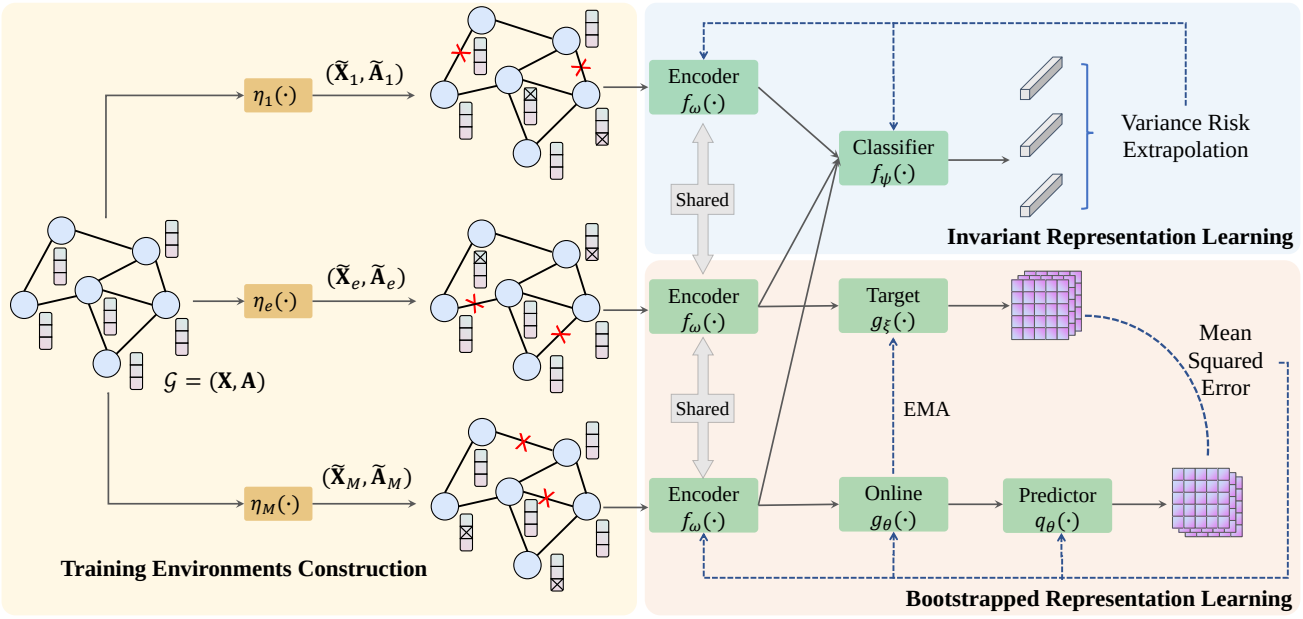


Figure 2: The framework of FLOOD on an example graph. The input graph is augmented to construct M training environments. The shared encoder outputs the representations of nodes in each environment and then forwards to the classifier. Variance risk extrapolation is applied among these environments to improve the generalization. The bootstrapped learning component trains the online encoder with the target to be the output of the target encoder. During the test phase, the parameters of the shared encoder are optimized under the self-supervised task on the test set.

To seek for good OOD generalization, we adopt the principle of Risk Extrapolation (REx). We aim to reduce the training risks but also increase the similarity of training risks. By encouraging equality of training risks, the risks are more likely to change less when the distribution shift happens at test time. Minimax-REx builds an affine combination of training risks with bounded coefficients as shown in Eq. (8). M is the number of environments, and the hyperparameter λ_{min} controls how much we extrapolate.

$$\begin{aligned}\mathcal{R}_{\text{MM-REx}}(\omega, \psi) &\doteq \max_{\substack{\sum_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{M} \lambda_e \mathcal{R}_e(\omega, \psi) \\ &= (1 - M\lambda_{\min}) \max_e \mathcal{R}_e(\omega, \psi) + \lambda_{\min} \sum_{e=1}^{M} \mathcal{R}_e(\omega, \psi)\end{aligned} \tag{8}$$

Practically, as there is a maximum in Eq. (8), it is hard and unstable to optimize $\mathcal{R}_{\text{MM-REx}}$. To tackle the problem, we simply replace the maximum with the variance of risks and obtain $\mathcal{R}_{\text{V-REx}}$ as Eq. (9) shows, where $\beta \in [0, +\infty)$ controls the balance between reducing average risk and enforcing equality of risks.

$$\mathcal{R}_{\text{V-REx}}(\omega, \psi) \doteq \beta \operatorname{Var}\left(\{\mathcal{R}_1(\omega, \psi), \ldots, \mathcal{R}_M(\omega, \psi)\}\right) + \sum_{e=1}^{M} \mathcal{R}_e(\omega, \psi) \tag{9}$$

3.3 Bootstrapped Representation Learning

As stated in Section 1, we design a self-supervised task to help the model fit the new test distribution. In the training phase, the self-supervised task will be trained jointly with the classification task with a shared graph encoder. In the test phase, we perform self-supervised learning on the test set to finetune the shared encoder for a better representation on the test set. To achieve this, the self-supervised task must be computationally efficient and capable of being trained within a few gradient descent steps. Although graph contrastive learning is widely utilized in self-supervised learning on graphs, it incurs a high computational and memory cost associated with negative sampling. Instead, we adopt Bootstrap Your Own Latent (BYOL) [5] in our framework, which learns to predict the target view from a different augmented view, obviating the need for negative samples.

Basically, BYOL produces two augmented views of the graph. We randomly choose two from Eq. (5), namely $\eta_i(\mathbf{X}, \mathbf{A})$ and $\eta_j(\mathbf{X}, \mathbf{A})$, $i, j = 1, \ldots, M$. As shown in Figure 2, two graph encoders with the same architecture but different weights are maintained for these two views. An online encoder g_θ is designed for the online view $\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{A}}_i\right)$ and a target encoder g_ξ is for the target view $\left(\widetilde{\mathbf{X}}_j, \widetilde{\mathbf{A}}_j\right)$, where θ and ξ denote two distinct sets of parameters.

The target encoder provides the regression targets to train the online encoder, and the target representation is denoted as $\mathbf{Z}_\xi = g_\xi\left(f_\omega\left(\widetilde{\mathbf{X}}_j, \widetilde{\mathbf{A}}_j\right)\right)$. The online representation $\mathbf{Z}_\theta = g_\theta\left(f_\omega\left(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{A}}_i\right)\right)$ is fed into a predictor q_θ that outputs a prediction of the target representation. The online parameter θ is optimized by minimizing the mean squared loss for the target prediction as Eq. (10) shows.

$$\begin{aligned}\mathcal{L}(\theta,\omega) &= \frac{1}{N}\sum_{k=1}^{N}\left\|\frac{q_\theta\left(\mathbf{Z}_{\theta,k}\right)}{\left\|q_\theta\left(\mathbf{Z}_{\theta,k}\right)\right\|_2}-\frac{\mathbf{Z}_{\xi,k}}{\left\|\mathbf{Z}_{\xi,k}\right\|_2}\right\|_2^2 \\ &= \frac{2}{N}\sum_{k=1}^{N}\left[1-\frac{\left\langle q_\theta\left(\mathbf{Z}_{\theta,k}\right),\mathbf{Z}_{\xi,k}\right\rangle}{\left\|q_\theta\left(\mathbf{Z}_{\theta,k}\right)\right\|_2\cdot\left\|\mathbf{Z}_{\xi,k}\right\|_2}\right].\end{aligned} \tag{10}$$

The target encoder is not updated by gradient descent and its parameters ξ are an exponential moving average (EMA) of the online parameters θ. More precisely, given a target decay rate $\tau \in [0,1]$, after each training step ξ are updated as Eq. (11) shows.

$$\xi \leftarrow \tau\xi + (1-\tau)\theta \tag{11}$$

In the training phase, both invariant learning and bootstrapped learning are jointly optimized under the overall loss as Eq. (12) shows, where α is a balanced parameter.

$$\min_{\theta,\omega,\psi}\mathcal{L}_{\text{train}} = \mathcal{L}(\theta,\omega) + \alpha\mathcal{R}_{\text{V-REx}}(\omega,\psi) \tag{12}$$

In the testing phase, δ steps of gradient descent are applied to the parameter of the shared graph encoder ω by minimizing the mean squared loss on the test graph $\mathcal{G}_{\text{test}}$ as Eq. (13) shows.

$$\min_{\omega}\mathcal{L}_{\text{test}} = \mathcal{L}(\theta,\omega)|\mathcal{G}_{\text{test}} \tag{13}$$

3.4 Overall Algorithm and Complexity Analysis

The overall training algorithm is summarized in Algorithm 1. Given a training graph $\mathcal{G}_{\text{train}}$ and a testing graph $\mathcal{G}_{\text{test}}$, we first construct M training environments by graph data augmentation (Line 1). With the GNNs initialized (Line 2), we get the node representation under each environment after the feedforward process (Line 5). The bootstrapped learning is conducted among two augmented views (Line 6). After that, the model is trained until convergence by minimizing Eq. (12) (Line 7), and the parameters of target encoder are updated by moving average (Line 8). During the test phase, we construct two views from $\mathcal{G}_{\text{test}}$ and perform bootstrapped learning for δ steps (Line 11). Finally, the shared encoder is updated by minimizing Eq. (13) (Line 11).

Consider a graph with N nodes and E edges, the average degree is $\bar{d}$. GNN with L layers compute embeddings in time and space $O(NL\bar{d}^2)$. FLOOD does $M+2$ encoder computations per update step (2 for target/online encoders, and M for each training environment) plus a prediction step. As there is no negative sampling, the overall time complexity is linear to the scale of the graph $O((NL\bar{d}^2)*(M+2))$.

Algorithm 1: FLOOD: Flexible invariant Learning framework for Out-Of-Distribution generalization on graphs

Input: $\mathcal{G}_{\text{train}}$: training graph, $\mathcal{G}_{\text{test}}$: testing graph, Y: Label of training nodes, M: Number of training environments, N_{epoch}: Number of training epochs, L: Number of GNN layers, τ: Moving average decay rate, δ: Test-time training steps, α, β: Balancing parameters for the loss.

Output: The prediction for $\mathcal{G}_{\text{test}}$.

1 Construct M training environments by Eq. (5);
2 Initialization parameters ω, ψ, and θ;
3 **repeat**
4 **for** $e = 1, \ldots, M$ **do**
5 Get the representation $\mathbf{h}_{e,i}^{(L)}$ for nodes of $\mathcal{G}_e$ w.r.t. Eq. (6);
6 Get $\mathbf{Z}_\theta$ and $\mathbf{Z}_\xi$ of $\mathcal{G}_{\text{train}}$ from two augmented views;
7 Train ω, ψ, and θ by minimizing Eq. (12);
8 Update ξ by exponential moving average Eq. (11);
9 **until** *Training epoch* $> N_{\text{epoch}}$;
10 Augment $\mathcal{G}_{\text{test}}$ and get $\mathbf{Z}_\theta$ and $\mathbf{Z}_\xi$ of $\mathcal{G}_{\text{test}}$;
11 Train ω by minimizing Eq. (13) and keep ψ and θ frozen;
12 return $f_\psi(f_\omega(\mathcal{G}_{\text{test}}))$.

4 EXPERIMENTS

In this section, we investigate the effectiveness of the proposed FLOOD model on both transductive and inductive graph OOD settings, with the aim of answering the following research questions.

- **RQ1**: Does FLOOD outperform the state-of-the-art methods for out-of-distribution generalizations on graphs?
- **RQ2**: How do the key components contribute to the results?
- **RQ3**: How does test-time training improve the generalization of GNNs?
- **RQ4**: What is the sensitivity of FLOOD with respect to the number of training environments and gradient descent steps during the test phase?

4.1 Experimental Setup

4.1.1 Datasets. We adopt four transductive node classification datasets from GOOD [6], a graph out-of-distribution benchmark, to verify the effectiveness of FLOOD in transductive settings. The inductive experiments are conducted on a multi-graph dataset named Twitch-explicit. The statistics of GOOD are shown in Table 1 and the statistics of Twitch-explicit is in Table 2.

Table 1: Statistics of GOOD datasets for transductive tasks.

Dataset	#Node	#Edge	#Class	#Feat	Domain
CBAS	700	3,962	4	4	Color
WebKB	617	1,138	5	1,703	University
Cora	19,793	126,842	70	8,710	Word/Degree
Arxiv	169,343	1,166,243	40	128	Time/Degree

Table 2: Statistics of Twitch-explicit for inductive tasks.

	DE	ES	FR	PTBR	RU	TW
Nodes	9,498	4,648	6,549	1,912	4,385	2,772
Edges	153,138	59,382	112,666	31,299	37,304	63,462
Density	0.003	0.006	0.005	0.017	0.004	0.017
Transit	0.047	0.084	0.054	0.131	0.049	0.057

Table 3: Performance Comparison for Transductive Node Classification under Covariate Shift

Dataset		CBAS		WebKB		GOOD-Cora				GOOD-Arxiv			
Domain		Color		University		Word		Degree		Time		Degree	
Covariate		ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD
Base	ERM	91.43	76.43	37.70	14.29	70.69	64.82	73.32	56.25	72.53	70.77	77.58	58.22
Invariant Learning	IRM	91.43	78.57	42.62	16.67	71.00	65.09	73.72	56.02	72.56	71.26	77.51	58.98
	VREx	92.86	78.57	36.07	16.67	70.79	64.77	73.32	56.28	72.58	71.25	77.73	58.95
	GroupDRO	92.86	80.00	42.62	14.29	70.74	64.82	73.32	56.37	72.61	71.14	77.63	59.09
Domain Generalization	DANN	94.29	74.29	42.62	15.25	70.69	64.77	73.32	56.20	72.48	71.16	77.38	59.19
	DeepCoral	91.43	78.57	39.34	16.67	70.69	64.80	73.32	56.34	72.76	71.28	77.75	59.20
Augmentation	Mixup	80.34	72.86	50.82	18.63	71.70	65.19	74.33	56.28	72.52	71.03	77.60	57.90
Graph OOD	SRGNN	87.14	71.43	42.62	13.32	70.14	64.32	71.00	53.88	72.25	70.77	76.05	57.66
	EERM	84.29	70.00	47.54	17.06	69.98	62.55	73.32	56.40	OOM	OOM	OOM	OOM
Ablation	FLOOD\Inv	90.26	75.78	42.31	15.34	70.23	63.23	73.23	56.32	72.41	70.21	77.42	56.82
	FLOOD\TtT	90.35	79.23	42.56	17.43	70.57	64.57	72.45	56.21	72.12	71.23	77.23	58.27
Ours	FLOOD	91.34	**83.53**	43.72	**18.95**	70.35	**66.23**	73.24	**56.64**	72.44	**72.13**	77.81	**59.47**

4.1.2 Compared Methods. We compare FLOOD with traditional invariant learning methods (IRM [1], VREx [13], GroupDRO [27]), domain generalization methods (DANN [4], DeepCoral [30]), graph data augmentation methods (Mixup [35]), and graph OOD methods (SRGNN [43], EERM [36]). Details are found in Appendix B.

We also derive two variants of FLOOD to analyze the performances of each component. They are **FLOOD\Inv** that removes invariant learning and performs test-time training on the GNN trained on the original graph, and **FLOOD\TtT** that removes test-time training and uses the invariant model to make a prediction.

4.1.3 Implementation Details. FLOOD is implemented in Pytorch 1.9.0 [21] with Python 3.8, and all the experiments run on a Ubuntu 18.04.1 server with 40 cores and 512GB memory. IRM, VREx, GroupDRO, DANN, and DeepCoral are based on the implementation of GOOD [6]. Mixup, SRGNN, and EERM are implemented using the source code provided by the authors. We report the average value of 5 runs for all the compared methods. For all compared methods and FLOOD, we adopt GCN as the backbone in the transductive settings, the same settings as GOOD does, for a fair comparison. And we adopt GAT as the backbone in the inductive settings, which is consistent with EERM.

For transductive datasets, GOOD has original splits for environment construction, and methods like IRM, VREx, GroupDRO, and DANN use the splits in the datasets. FLOOD further augments the environments with graph data augmentation methods. Different from that, the inductive dataset Twitch does not have an environment split thus all the compared methods share the same environments as done by FLOOD.

4.1.4 Hyper-parameter Settings. The parameters of FLOOD are optimized with RMSprop [26] optimizer with learning rate lr=1e-3. Hyper-parameter tuning is conducted using grid search for most methods. In FLOOD, $N_{\text{epoch}} = 300$, $\tau = 0.99$, $L = 3$, $\delta = 10$, $\alpha = 1$, $\beta = 1$. For GOOD-CBAS, $M = 10$ for both covariate shift and concept shift. For other datasets in transductive tasks, $M = 10$ for covariate shift, and $M = 3$ for concept shift. For Twitch, $M = 6$ for inductive tasks.

4.1.5 Metrics. For transductive node classification, we adopt **Accuracy** to measure the performance of all the compared methods since all the benchmarks are multi-class classifications. For inductive node classification, we adopt both **AUC** and **Accuracy** to measure the performance since it is a binary classification. For both, higher scores indicate better performance of an approach.

4.2 Performance Comparison (RQ1)

To answer RQ1, we evaluate the performance of FLOOD on both transductive and inductive node classification tasks. The scores of IID and OOD are reported in Table 3-5. We have the following observations.

Firstly, FLOOD outperforms current state-of-the-art methods due to its flexibility during the test phase. EERM achieves the SOTA performance for multi-graph generalization in Table 5. Although FLOOD does not optimize the generator for environment generation, fine-tuning the parameters of the encoder can also improve the generalization ability on the target distribution and achieve the best performance. In contrast, EERM does not achieve the second-best performance for transductive tasks in Table 3 and Table 4. This is because the environments used in EERM are generated by trained generators for larger variance, whereas other methods utilize the split provided by the dataset, which is closely related to the spurious domain. Our environmental construction with data augmentation can outperform the trained generator as well.

Secondly, the compared methods exhibit different performances under different distribution shifts. Traditional invariant methods like IRM and VREx work well for both covariate shift and concept shift, but cannot handle the inductive node classification. The

Table 4: Performance Comparison for Transductive Node Classification under Concept Shift

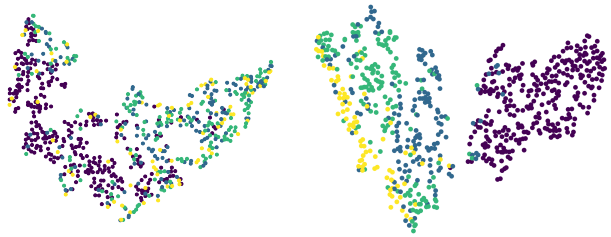
Dataset		CBAS		WebKB		GOOD-Cora				GOOD-Arxiv			
Domain		Color		University		Word		Degree		Time		Degree	
Concept		ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD
Base	ERM	90.00	81.43	63.33	26.61	65.90	64.35	69.00	61.09	74.86	67.35	75.06	62.29
Invariant Learning	IRM	90.71	82.52	61.67	27.23	65.96	64.40	68.04	61.23	74.37	67.40	75.38	62.49
	VREx	90.00	82.14	63.33	28.44	65.90	64.37	68.93	61.10	74.74	67.29	74.96	62.72
	GroupDRO	89.29	83.57	63.33	29.92	66.09	64.49	68.87	61.12	74.51	67.47	75.22	62.63
Domain Generalization	DANN	90.00	82.71	63.33	26.61	65.83	64.53	68.93	61.03	74.76	67.03	74.91	62.55
	DeepCoral	90.00	81.43	63.21	28.42	66.09	64.49	69.13	61.14	74.82	67.62	75.07	62.49
Augmentation	Mixup	93.57	64.29	63.33	30.28	70.58	64.77	70.15	63.12	74.74	65.17	72.28	60.10
Graph OOD	SRGNN	90.00	80.71	68.33	25.69	65.96	65.20	69.26	60.62	74.56	67.15	74.81	62.07
	EERM	81.43	62.14	63.33	26.53	65.06	62.66	65.85	58.23	OOM	OOM	OOM	OOM
Ablation	FLOOD\Inv	90.25	82.35	63.24	25.23	65.32	64.24	68.34	61.24	74.32	67.21	74.21	62.34
	FLOOD\TtT	90.32	82.31	63.56	28.35	65.82	64.46	68.23	61.23	74.21	67.12	74.23	62.52
Ours	FLOOD	90.47	**84.25**	63.72	**31.95**	65.85	**65.23**	68.24	**63.64**	74.24	**67.93**	74.81	**63.47**

Table 5: Performance Comparison for Inductive Node Classification under Distribution Shift

Dataset		ES		FR		PTBR		RU		TW	
Metric		AUC	Acc	AUC	Acc	AUC	Acc	AUC	Acc	AUC	Acc
ID		68.23	58.47	68.69	66.75	69.12	65.61	64.76	63.60	64.81	62.48
Base	ERM	62.10	45.59	62.12	42.41	62.60	50.71	50.25	38.52	51.29	40.95
Invariant Learning	IRM	63.50	49.97	62.74	43.53	63.69	51.85	55.19	39.33	51.90	41.47
	VREx	64.53	43.98	62.37	46.02	64.20	51.77	52.76	41.29	54.80	42.40
	GroupDRO	64.78	50.76	62.48	46.56	64.22	51.91	55.60	43.41	54.72	40.79
Domain Generalization	DANN	62.20	43.19	62.62	46.69	64.55	47.03	55.50	44.32	54.19	41.59
	DeepCoral	63.03	43.61	62.75	46.53	64.73	47.92	55.75	44.63	54.82	42.39
Augmentation	Mixup	62.28	47.07	60.95	40.92	61.73	46.81	54.76	35.63	56.98	44.24
Graph OOD	SRGNN	63.30	42.72	60.38	43.65	60.69	54.28	54.53	41.04	55.45	42.11
	EERM	65.18	51.74	63.04	46.86	64.91	51.49	56.68	44.91	58.77	46.07
Ablation	FLOOD\Inv	63.63	42.74	62.36	43.21	62.12	51.43	52.52	40.21	52.37	40.21
	FLOOD\TtT	64.32	43.84	63.45	46.23	64.23	52.32	53.25	42.53	55.21	42.93
Ours	FLOOD	**66.77**	**54.95**	**65.48**	**48.66**	**65.59**	**56.98**	**57.13**	**45.80**	**59.93**	**48.99**

reason is that the split is based on the spurious domain for the transductive datasets and seeking similar performance across various environments would help with the generalization. However, for the inductive dataset, the shift domain is unknown and the environments are from data augmentation. Invariant learning methods rely on environment construction and thus perform worse than those regularization-based methods like DANN and SRGNN. The bootstrapped learning component in FLOOD, therefore, leads to better generalization performance on inductive tasks compared to transductive tasks.

Finally, data augmentation methods can be helpful for OOD generalization in some cases but they can also degrade the performance if not used properly. For the results of covariate shift in Table 3, Mixup can achieve the second-best results for GOOD-WebKB and GOOD-Cora-Word. The results indicate that data augmentation can enrich the training distribution for better generalization. In contrast, for GOOD-CBAS and GOOD-Arxiv-Degree, the effectiveness of Mixup is negative. The possible reason is that the augmented views are spuriously correlated to the shift domain like degree, making the generalization even worse than ERM. FLOOD can mitigate the limitation of data augmentation because it can flexibly adapt the encoder to the target distribution during the test phase.



(a) FLOOD w/o TtT (b) FLOOD w TtT

Figure 3: Visualization of FLOOD on GOOD-CBAS without and with test-time training.

4.3 Ablation Study (RQ2)

To answer RQ2, we evaluate two key modules of FLOOD, namely invariant representation learning and bootstrapped representation learning, by removing each module respectively. We report the ablation results of transductive tasks in Table 3 and 4 and the results of inductive tasks are in Table 5. The full model FLOOD always achieves the best scores compared with the two variants, FLOOD\Inv and FLOOD\TtT, indicating that each module is necessary for OOD generalization.

Removing the bootstrapped learning part, FLOOD\TtT does not have test-time training steps during the test phase and achieves comparable performance to that of VREx or GroupDRO. The performance of FLOOD\TtT declines larger in the inductive tasks than that in the transductive tasks, which indicates that test-time training contributes more when the spurious domain is unavailable such as the multi-graph generalization.

Without the invariant learning part, FLOOD\Inv just finetunes the GNN encoder by BYOL during the test phase and the results show that the performance degradation is even larger than that of FLOOD\TtT. We conclude that combining test-time training with invariant learning contributes more to the improvement of generalization.

4.4 Visualization (RQ3)

To answer RQ3, we visualize the output of the shared encoder on GOOD-CBAS without and with the test-time training in Figure 3 (The color indicates the node label). It is shown that the shared encoder fine-tuned by FLOOD learns more discriminative representations, thanks to the bootstrapped learning during the test phase. These highly discriminative representations potentially help to improve the generalization ability of the classifier on the test set than the less discriminative ones without test-time training.

4.5 Sensitivity Analysis (RQ4)

To answer RQ4, we further evaluate the sensitivity of FLOOD with respect to the number of training environments and the number of gradient steps during the test phase. Due to the limited space, we only report the results of GOOD-Cora and GOOD-Arxiv for the transductive tasks for better visualization. For the inductive tasks, we only report ES, FR, and PTBR for demonstration. Other datasets also exhibit similar trends of sensitivity.

The number of training environments is closely related to risk extrapolation, which has been defined in Section 3.2. We evaluate the sensitivity of FLOOD on GOOD-Cora and GOOD-Arxiv under covariate shift in Figure 4(a) and concept shift in Figure 4(b). GOOD provides 10 training environments for covariate shift and 3 environments for concept shift, thus we set the range to be 2 to 10 with step 2 for covariate shift and 1 to 5 with step 1 for concept shift[1]. The sensitivity on the inductive task is in Figure 4(c) and the range is 2 to 10 with step 2 as well.

For the covariate shift and concept shift, the number of environments to achieve the best performance equal the provided number in the GOOD benchmark, 10 and 3, respectively. As demonstrated in Figure 4(b), increasing the number of training environments does not gain much improvement, while decreasing the number would hurt the performance. Because a smaller number of training environments may not cover the extrapolated distribution for generalization. For inductive tasks, the optimal number is 6, and differently, increasing the number in inductive tasks would hurt the performance a lot. This is because the distribution shift in inductive tasks is not related to a single spurious domain. Too many environments will make it harder to learn a good invariant representation.

The number of gradient steps during the test phase decides how much we update the shared encoder to adapt to the test distribution. As shown in Figure 4(d) to Figure 4(e), the performance improves as the number of gradient steps increases, reaches a peak, and then begins to decline. The best performance is achieved at around 10 for all three settings. If the number of gradient descent exceeds 30, though the shared encoder may fit the target distribution, the classifier trained on the training environments will not perform well, as the over-updated representations will deviate from the learned latent space.

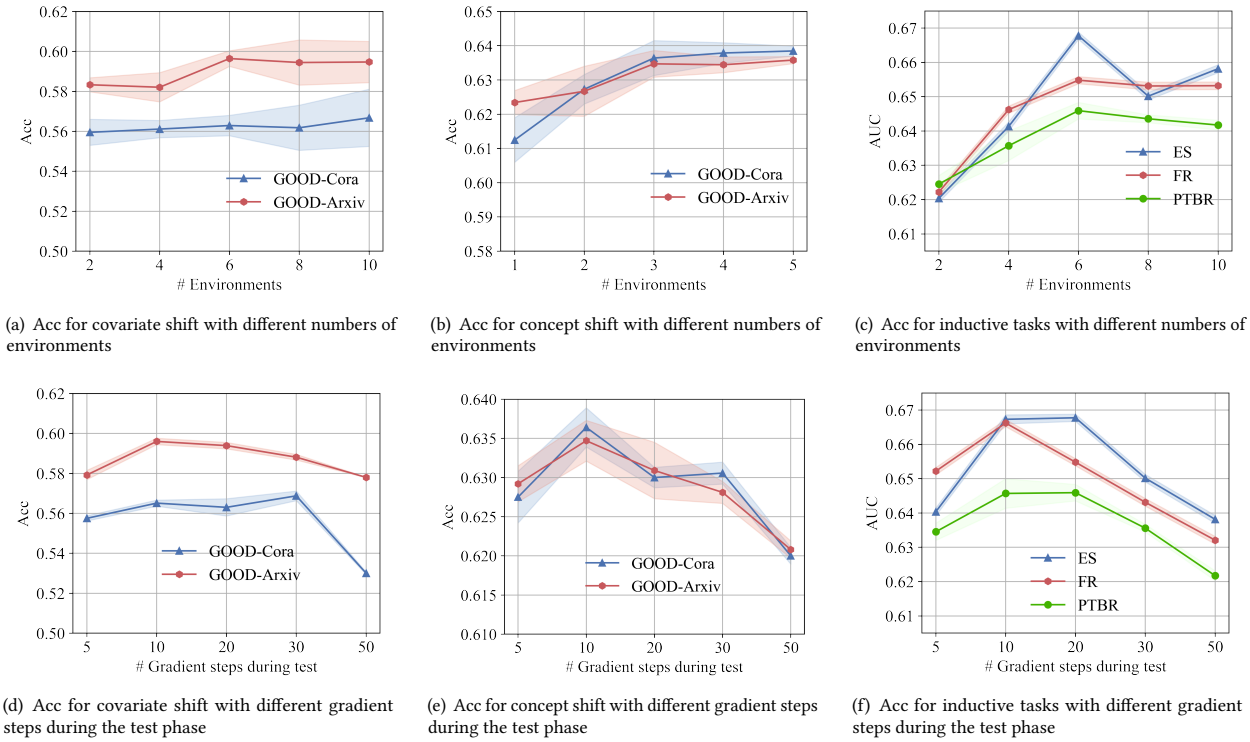
5 RELATED WORK

Out-of-distribution Generalization. Out-of-Distribution (OOD) generalization problem addresses the challenging setting where the testing distribution is unknown and different from that of the training. Several kinds of strategies can be applied to tackle OOD generalization [28]. Causal inference aims to learn variables in the causal graph in an unsupervised or semi-supervised way. With the learned causal representation, one can capture the latent data generation process, which can help to resist the distributional shifts induced by interventions. Peters et al. [22] first try to investigate the fact that "invariance" could infer the causal structure under necessary conditions and propose Invariant Causal Prediction (ICP).

Deriving from causal inference-based methods, invariant learning methods, typified by invariant risk minimization(IRM, [1]), target on latent causal mechanisms and extend ICP to more practical settings. VREx [13] encourages reducing training risks while increasing the similarity of training risks under the variance risk extrapolation paradigm. GroupDRO [27] learns models that minimize the worst-case training loss over a set of pre-defined groups.

Test-time training [20, 31] is an emerging paradigm to solve OOD generalization problems. It trains the model on both the main task and the self-supervised learning task, and updates the model based only on the SSL task at test time.

[1]We perform different data augmentation on the three given environments to construct five environments for evaluation.



(a) Acc for covariate shift with different numbers of environments

(b) Acc for concept shift with different numbers of environments

(c) Acc for inductive tasks with different numbers of environments

(d) Acc for covariate shift with different gradient steps during the test phase

(e) Acc for concept shift with different gradient steps during the test phase

(f) Acc for inductive tasks with different gradient steps during the test phase

Figure 4: Sensitivity of FLOOD w.r.t the number of training environments (a-c) and the number of gradient steps during the test phase (d-f). The marker point represents the mean value of 5 runs, and the shaded area corresponds to the standard deviation.

The methods mentioned above improve the generalization ability of current models for out-of-distribution generalization. Different from them, our major concern in this work is the out-of-distribution problem on graphs. The principle is applicable but the environment is more complicated due to the complex structure of the graphs.

Out-of-distribution Generalization on Graphs. Graph OOD generalization can be more complex because distribution shifts on graphs can appear in a variety of forms such as attributes and structures, making it difficult to identify the invariance. Moreover, environment construction or inference, which are often required by OOD methods on Euclidean data, can be highly expensive to obtain for graphs due to the structural relationship of the nodes.

Zhou et al. [42] theoretically studied the ability of GNNs to achieve counterfactually-invariant representations for inductive OOD link prediction tasks. GIL [14] can capture the invariant relationships between predictive graph structural information and labels in a mixture of latent environments by jointly optimizing three tailored modules. DIR [38] discovers causal rationales that are invariant across different distributions to improve generalization. CIGA [3] characterizes distribution shifts with causal models. MoleOOD [39] enhances the robustness of molecule learning and can infer the environment in a data-driven manner. SR-GNN [43] is designed to account for distributional differences between biased training data and a graph's true inference distribution. EERM [36] trains multiple context generators to maximize the variance of risks from multiple virtual environments so the model can explore to extrapolate from a single observed environment.

GIL, DIR, MoleOOD, and gMPNN investigate different tasks for OOD generalization on graphs like graph classification and link prediction. Different from them, we consider the OOD problem of node-level tasks on graphs. EERM and SRGNN can be applied to node-level tasks but FLOOD is more flexible since its encoder can be refined for better generalization during the test phase.

6 CONCLUSION

In this study, we investigated the issue of out-of-distribution (OOD) generalization in graph representation learning and proposed a new solution, FLOOD, which combines invariant representation learning and bootstrapped representation learning. Our method aims to find a balance between stability across different training environments and adaptability to the test distribution. By using invariant representation learning, FLOOD learns a generalized representation that is robust to variations in the training environment. The bootstrapped representation learning component then allows the model to further adapt to the test distribution by conducting self-supervised learning during the test phase.

ACKNOWLEDGMENTS

The research work is supported by National Key R&D Plan No. 2022YFC3303302, National Natural Science Foundation of China under Grant (No.61976204). Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS, Beijing Nova Program Z201100006820062.

REFERENCES

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* (2019).

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Yongqiang Chen, Yonggang Zhang, Yatao Bian, Han Yang, MA KAILI, Binghui Xie, Tongliang Liu, Bo Han, and James Cheng. 2022. Learning Causally Invariant Representations for Out-of-Distribution Generalization on Graphs. In *Advances in Neural Information Processing Systems*.

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[5] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33 (2020), 21271–21284.

[6] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. 2022. Good: A graph out-of-distribution benchmark. *Advances in Neural Information Processing Systems* (2022).

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[8] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1857–1867.

[10] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Auc-oriented graph neural network for fraud detection. In *Proceedings of the ACM Web Conference 2022*. 1311–1321.

[11] Yuanfeng Ji, Lu Zhang, Jiaxiang Wu, Bingzhe Wu, Long-Kai Huang, Tingyang Xu, Yu Rong, Lanqing Li, Jie Ren, Ding Xue, et al. 2022. DrugOOD: Out-of-Distribution (OOD) Dataset Curator and Benchmark for AI-aided Drug Discovery–A Focus on Affinity Prediction Problems with Noise Annotations. *arXiv preprint arXiv:2201.09637* (2022).

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[13] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. 2021. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*. PMLR, 5815–5826.

[14] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Learning invariant graph representations for out-of-distribution generalization. *Advances in Neural Information Processing Systems* (2022).

[15] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 925–935.

[16] Kuan Li, Yang Liu, Xiang Ao, and Qing He. 2023. Revisiting Graph Adversarial Attack and Defense From a Data Distribution Perspective. In *The Eleventh International Conference on Learning Representations*.

[17] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. 2022. Pre-training molecular graph representation with 3d geometry. In *International Conference on Learning Representations*.

[18] Yang Liu, Xiang Ao, Fuli Feng, and Qing He. 2022. UD-GNN: Uncertainty-aware debiased training on semi-homophilous graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1131–1140.

[19] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*. 3168–3177.

[20] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems* 34 (2021), 21808–21820.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

[22] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. 2016. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* (2016), 947–1012.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

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

[25] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[26] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[27] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. 2020. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*.

[28] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624* (2021).

[29] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*.

[30] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 443–450.

[31] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*. PMLR, 9229–9248.

[32] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*. PMLR, 21076–21089.

[33] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[35] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2021. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*. 3663–3674.

[36] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. 2022. Handling Distribution Shifts on Graphs: An Invariance Perspective. In *International Conference on Learning Representations*.

[37] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[38] Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. 2022. Discovering invariant rationales for graph neural networks. In *International Conference on Learning Representations*.

[39] Nianzu Yang, Kaipeng Zeng, Qitian Wu, Xiaosong Jia, and Junchi Yan. 2022. Learning substructure invariance for out-of-distribution molecular representations. In *Advances in Neural Information Processing Systems*.

[40] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019).

[41] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[42] Yangze Zhou, Gitta Kutyniok, and Bruno Ribeiro. 2022. OOD Link Prediction Generalization Capabilities of Message-Passing GNNs in Larger Test Graphs. *Advances in Neural Information Processing Systems* (2022).

[43] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. 2021. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems* 34 (2021), 27965–27977.

A DATASETS

We adopt four transductive node classification datasets from the graph OOD benchmark GOOD [6] to verify the effectiveness of FLOOD in transductive settings. The inductive experiments are conducted on a multi-graph dataset named Twitch-explicit.

GOOD-CBAS is a synthetic dataset modified from BA-Shapes [40]. The input is a graph created by attaching 80 house-like motifs to a 300-node Barabási–Albert base graph and the task is to predict the role of nodes, including the top/middle/bottom node of a house-like motif or the node from the base graph, forming a 4-class classification task. Different from BA-Shapes, CBAS includes colored features as spurious features that OOD algorithms need to tackle node color differences in covariate splits and color-label correlations in concept splits.

GOOD-WebKB is a graph of university webpage. Nodes are university web pages and edges are hyperlinks between them. Node features are bag-of-words representations of the webpages. Node labels are one of student, project, course, staff, or faculty. The OOD split is conducted based on the university domain but the classification relies on the webpage.

GOOD-Cora is a citation network labeled on the paper topic. The input is a small-scale citation network graph, in which nodes represent scientific publications and edges are citation links. The task is a 70-class classification of publication types. The OOD splits are based on two domains, namely, word and degree. The first one is the word diversity defined by the selected-wordcount of a publication, purely irrelevant to the label. The second one is the node degree in the graph, implying that the popularity of a paper should not determine the class of a paper.

GOOD-Arxiv is the citation network between all Computer Science (CS) arXiv papers indexed by MAG [33]. Each node is an arXiv paper, and each directed edge indicates that one paper cites another one. Each paper comes with a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. The task is to predict the 40 subject areas of arXiv CS papers, e.g., cs.AI, cs.LG, and cs.OS, which are manually labeled by the paper's authors and arXiv moderators. The OOD splits are based on two domains, namely, time and degree.

Twitch-explicit [25] contains 7 networks where Twitch users are nodes, and mutual friendships between them are edges. Node features are games liked, location, and streaming habits. Each graph is associated with users of a particular region. The class labels denote whether a streamer uses explicit language. The model is trained on DE, valid on EN, and tested on the other 5 graphs.

B COMPARED METHODS

- **ERM** [12]: graph convolution network optimized by empirical risk minimization.
- **IRM** [1]: includes a regularized objective enforcing simultaneous optimality of the same classifier in all environments.
- **VREx** [13]: encourages reducing training risks while increasing the similarity of training risks under variance risk extrapolation paradigm.
- **GroupDRO** [27]: learn models that minimize the worst-case training loss over a set of pre-defined groups.
- **DANN** [4]: adversarially trains the regular classifier and a domain classifier to make features indistinguishable.
- **DeepCoral** [30]: encourages features in different domains to be similar by minimizing the deviation of covariant matrices from different domains.
- **Mixup** [35]: a two-branch Mixup graph convolution to interpolate the irregular graph topology.
- **SRGNN** [43]: includes both a regularization for addressing distribution shift in learnable layers and an instance reweighting component which is capable of handling situations where a graph inductive bias is added after feature encoding.
- **EERM** [36]: includes multiple context generators that are adversarially trained to maximize the variance of risks from multiple virtual environments and applies invariant learning on these environments.