DeCaf: A Causal Decoupling Framework for OOD Generalization on Node Classification

Xiaoxue Han
Stevens Institute of Technology

Huzefa Rangwala
George Mason University

Yue Ning
Stevens Institute of Technology

Abstract

Graph Neural Networks (GNNs) are susceptible to distribution shifts, creating vulnerability and security issues in critical domains. There is a pressing need to enhance the generalizability of GNNs on out-of-distribution (OOD) test data. Existing methods that target learning an invariant *(feature, structure)-label* mapping often depend on oversimplified assumptions about the data generation process, which do not adequately reflect the actual dynamics of distribution shifts in graphs. In this paper, we introduce a more realistic graph data generation model using Structural Causal Models (SCMs), allowing us to redefine distribution shifts by pinpointing their origins within the generation process. Building on this, we propose a **ca**sual **de**coupling **f**ramework, **DeCaf**, that independently learns unbiased *feature-label* and *structure-label* mappings. We provide a detailed theoretical framework that shows how our approach can effectively mitigate the impact of various distribution shifts. We evaluate **DeCaf** across both real-world and synthetic datasets that demonstrate different patterns of shifts, confirming its efficacy in enhancing the generalizability of GNNs. Our code is available at: `https://github.com/hanxiaoxue114/DeCaf-GraphOOD`.

1 INTRODUCTION

Graph Neural Networks (GNNs) perform node classification tasks by learning the relationships between node features, local structures, and node labels on a training graph. They have demonstrated promising performance across various applications, such as social recommendation, traffic forecasting, and chemical property prediction [Wu et al., 2021]. However, their success largely relies on the in-distribution (ID) assumption [Liu et al., 2023]. In real life, test samples are often collected from different distributions such as various geographical areas, domains, or time periods, leading to potentially different and unknown distributions from the training data. Consequently, the model might learn an incorrect mapping of *(feature, structure)-label* relationships that fail on test data, leading to a well-known out-of-distribution (OOD) problem [Arjovsky et al., 2020].

One *de facto* approach to graph OOD generalization [Bai et al., 2021, Krueger et al., 2020, Sagawa et al., 2019, Shen et al., 2021, Ye et al., 2021] assumes that there exist "true" correlations between input and labels that are invariant under distribution shifts. These correlations can be obtained by identifying the non-causal/spurious parts of the input (*e.g.* sub-ego-networks or subvectors of the node's embedding). However, they model the features of a node and its neighborhood as a single entity. By doing so, they implicitly assume that the shifts in features ($\mathbf{X}$) and structures ($\mathcal{A}$) occur simultaneously and cannot be separated from one another. A detailed discussion on the limitations of existing methods is provided in Section 4.

However, we claim that such an assumption may fail on many real-world graphs (*e.g.*, citation graphs, social networks). The generation process of graph data is complex: for a node, its features and local topology can be viewed as reflections of its true representations by different "observers". For example, to predict whether someone might be prone to drug addiction based on their online social networks, we can analyze their profile features (*e.g.*, job, social status, photos) as the public image they choose to present. The accounts they follow can indicate additional aspects (*e.g.*, hobbies, personal life, mental status) not explicitly stated in their profiles. Both the profile features and their

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

network connections provide insights into the user's true status, but they highlight different facets through distinct mappings. Together, they offer valuable information for predicting the target behavior. However, these elements may display varying distribution patterns over time or across different locations. For instance, individuals might disclose different profile information depending on state laws, or their network connections might change following updates to the social network's recommendation algorithm. Such variations can occur separately and both impact the original *(feature, structure)-label* relationships.

We formalize the graph generation process as a Structural Causal Model (SCM). Based on the graph generation process, we redefine the types of distribution shifts. Previously, graph OOD largely followed definitions from general data: for *covariate shift*, $P^{\text{train}}(\mathbf{X}, \mathcal{A}) \neq P^{\text{test}}(\mathbf{X}, \mathcal{A})$; for concept shift, $P^{\text{train}}(\mathbf{Y}|\mathbf{X}, \mathcal{A}) \neq P^{\text{test}}(\mathbf{Y}|\mathbf{X}, \mathcal{A})$ [Gui et al., 2022]. We reformulate the definition, attributing *covariate shift* to changes in the true representations, and define two types of *concept shift* based on the different mappings of features and structure, respectively. We justify the universality of our new definitions.

To this end, we demonstrate that any type of distribution shift can alter the *(feature, structure)-label* mapping. However, we observe that because the generating mechanism of the features or structures does not change, the true *feature-label* or *structure-label* mappings should remain invariant. Based on this observation, we propose a causal decoupling framework, **DeCaf**, that learns unbiased *feature-label* or *structure-label* mappings as causal effects for predicting node labels. Intuitively, **DeCaf** answers the question: *"What information about the label can the node features provide when its local structure is unavailable, and vice versa?"* We provide a theoretical analysis to demonstrate the feasibility and effectiveness of **DeCaf**. We also present an implementable paradigm of **DeCaf** that utilizes Generalized Robinson Decomposition to estimate the causal effects as a practical solution. We evaluate our proposed method across both real-world and synthetic datasets with different patterns of shifts. The results consistently demonstrate that our proposed method improves the generalization ability of GNNs on the node classification task.

2 CAUSAL DECOUPLING FRAMEWORK

In this section, we introduce the causal decoupling framework, **DeCaf**, which performs node classification by independently estimating the treatment effects of node features and neighborhood representations.

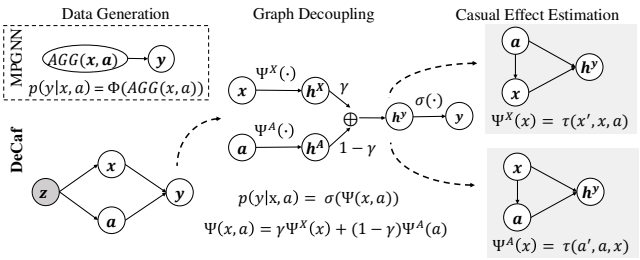


Figure 1: An overview of the conceptual flow. First, we introduce a new data generation process with the SCM. We separately estimate the individual impact of the node features and neighborhood representations on node labels in graph decoupling. To achieve an unbiased estimation of their impact, we propose to treat the impact as a treatment effect, which can be estimated with a casual estimation model that considers the confounding effect.

We develop a comprehensive theoretical foundation to support the rationale: Initially, we present a novel **graph generation process** using a Structural Causal Model (SCM), which contrasts with the assumptions underlying most GNN models (section 2.1). Utilizing this SCM, we redefine various **types of distribution shifts in graph data** and analyze how each SCM element changes under these shifts (Section 2.2). This analysis leads to the development of the **causal decoupling framework**, which seeks to separately assess the direct impacts of node features and neighborhood representations on node labels (section 2.3). To achieve an unbiased estimation of their impact, we treat the impact as a treatment effect, which can be estimated with a **casual estimation** model that considers the confounder effect (section 2.4). We visually summarize this conceptual flow in Figure 1. Following this theoretical basis, we propose **a practical end-to-end paradigm** that leverages SOTA casual inference technologies to estimate the decoupled representations and make final predictions, as detailed in Section 2.4.

2.1 Graph Generation Process

We build a SCM for graph generation processes based on the assumption that for each node v_i, there exists an unobserved raw latent vector $\mathbf{z}_i \in \mathbb{R}^p$ that is fully informative about its *"true nature"*, and its ground-truth label, $\mathbf{y}_i \in \mathbb{R}^k$, is an affine transformation of $\mathbf{z}_i$ to the label space. For a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set, we directly observe node features $\mathbf{X} \in \mathbb{R}^{n \times d}$, and the adjacency matrix $\mathcal{A} \in \{0, 1\}^{n \times n}$. Although the latent variable matrix $\mathbf{Z} \in \mathbb{R}^{n \times p}$ cannot be directly observed, we believe there exists a "*hidden observer*" that can observe $\mathbf{Z}$ and decide $\mathbf{X}$ and $\mathcal{A}$ based on some rules.

Definition 1. *The "true nature", denoted as $\boldsymbol{z}$, is an unobserved latent variable representing "all facts" (extrinsic or intrinsic) about a node in a graph. For a node instance v_i, knowing $\boldsymbol{z}_i$ provides sufficient information about its features $\boldsymbol{f}_i$ and its connectivity with any other node u_j as $\mathcal{A}_{ij}$. Specifically, there exists a function such that $\mathbf{x}_i = f(\mathbf{z}_i)$ and a function such that $\mathcal{A}_{ij} = g(\mathbf{z}_i, \mathbf{z}_j)$.*

An intuitive example. A person's "*true nature*" would encompass not only extrinsic details like gender and education but also intrinsic qualities such as personality and beliefs. These intrinsic features are difficult to measure directly and completely, yet they largely influence a person's public profile (*e.g.*, personal webpage) and social relationships (*e.g.*, friendships). For example, the content of a personal webpage is shaped not only by true experiences but also by the individual's personality, which affects how those experiences are presented. Understanding a person's *true nature* provides a complete view of their behaviors.

For simplicity of analysis, we assume the features of node v_i, $\mathbf{x}_i$, is an affine transformation of z_i, and $\mathcal{A}_{ij}$ is decided by the similarity between some linear transformations of $\mathbf{z}_i$ and $\mathbf{z}_j$.

Assumption 1. *The generation process of $(\mathbf{X}, \mathbf{Y}, \mathcal{A})$ given $\mathbf{Z}$ can be expressed as follows:*

$$\mathbf{x}_i = \mathcal{M}_f \mathbf{z}_i + \mathbf{b}_f, \tag{1}$$

$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y, \tag{2}$$

$$p(\mathcal{A}_{ij} = 1) = c \cdot \left(\|\mathcal{M}_s \mathbf{z}_i, \mathcal{M}_o \mathbf{z}_j\|_2^2 + 1 \right)^{-1}, \tag{3}$$

where $\mathcal{M}_f \in \mathbb{Z}^{d \times p}$, $\mathcal{M}_y \in \mathbb{Z}^{k \times p}$, $\mathcal{M}_s \in \mathbb{Z}^{q \times p}$, and $\mathcal{M}_o \in \mathbb{Z}^{q \times p}$ are constant linear transformation matrices, $\mathbf{b}_f \in \mathbb{R}^d$ and $\mathbf{b}_y \in \mathbb{R}^k$ are constant vectors, $\| \cdot \|_2^2$ is euclidean distance, and $0 \leq c \leq 1$ is a constant number to control the density of the adjacency matrix.

We claim that Assumption 1 can be generalized to different scenarios. For instance, a homophilous graph would have $\mathcal{M}_o$ $\mathcal{M}_s$ assigned with the same values as $\mathcal{M}_y$; conversely, a heterophilous graph would have $\mathcal{M}_o$ and $\mathcal{M}_s$ to be opposite with each other; when $\mathcal{M}_o$ or $\mathcal{M}_s$ are assigned with values close to zeros, the connection will show more randomized behaviors.

According to Assumption 1, node v_i's feature $\mathbf{x}_i$ is only dependent on $\mathbf{z}_i$; however, its connection with other nodes depends on the latent variable $\mathbf{Z}$ of the whole graph, and it seemingly brings *spill-over effects/inferences*, which occurs when the treatment received by one instance affects the outcome of another instance. This effect breaks *Stable Unit Treatment Value Assumption (SUTVA)* that the potential outcomes of any unit do not vary with the treatment assigned to other units. It is one of the core assumptions

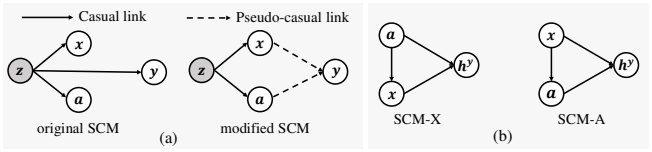


Figure 2: (a) SCMs represent a node's data generation process. (a-*left*) is the original SCM, and (a-*right*) is the modified SCM by replacing the causal link between z and y with pseudo-casual links. (b) The two-view SCMs for the casual effect estimation. For SCM-X, $\mathbf{a}$ is the treatment, and $\mathbf{x}$ is the confounder. For SCM-A, the treatment and the confounder are reversed.

in causal inference as we will discuss in section 2.4. To address this, we investigate the correlation between the features of a central node and its neighboring nodes. As the adjacent matrix $\mathcal{A}$ does not contain the features of neighboring nodes, we define an embedding matrix

$$\mathbf{a}_i = \text{agg}(\{\mathcal{M}_a \mathbf{z}_j\}) \quad \text{where} \quad j \in \mathcal{N}_i^1 \cup \mathcal{N}_i^2 ... \mathcal{N}_i^l, \tag{4}$$

where $\mathbf{a}_i$ is the i-th vector of $\mathcal{A}$ which represents the neighborhood information of all nodes, $\mathcal{M}_a$ is a linear transformation matrix, and $\mathcal{N}_i^l$ is the node set of l-hop neighbors of node v_i. agg$(\cdot)$ is an aggregation function (e.g. mean). Although this may introduce undesired correlations between samples that complicate our analysis, we show that under the *Law of Large Numbers*, $\mathbf{a}_i$ experiences negligible spillover effects from other samples. Further details are discussed in Appendix D.

To this end, we build a SCM in Figure 2 (a-*left*) to represent the relationships between the variables $\mathbf{z}$, $\mathbf{x}$, $\mathbf{a}$, and $\mathbf{y}$. When developing a machine learning model, the *Close World Assumption* is generally followed, that the training data encompasses sufficient information to make accurate predictions. Applying that to our case, given the unobservability of $\mathbf{z}$, we need to assume that all features within $\mathbf{z}$ pertinent to $\mathbf{y}$ are recoverable through $\mathbf{x}$ and $\mathbf{a}$. For the convenience of our narrative, we assume $\mathbf{z}$ is directly caused by $\mathbf{a}$ and $\mathbf{x}$, and we replace the causal link between $\mathbf{z}$ and $\mathbf{y}$ with pseudo-causal links $\mathbf{x} \dashrightarrow \mathbf{y}$ and $\mathbf{a} \dashrightarrow \mathbf{y}$. **Future analysis is based on the modified SCM in Figure 2 (a-*right*).** Based on the modified SCM, the correlation between $\mathbf{x}$ and $\mathbf{y}$ is constituted by two back door paths: $\mathbf{x} \leftarrow \mathbf{z} \rightarrow \mathbf{y}$ and $\mathbf{x} \leftarrow \mathbf{z} \rightarrow \mathbf{a} \rightarrow \mathbf{y}$. Similarly, the correlation between $\mathbf{a}$ and $\mathbf{y}$ is constituted by $\mathbf{a} \leftarrow \mathbf{z} \rightarrow \mathbf{y}$, $\mathbf{a} \leftarrow \mathbf{z} \rightarrow \mathbf{x} \rightarrow \mathbf{y}$. For the paths $\mathbf{x} \leftarrow \mathbf{z} \rightarrow \mathbf{y}$ and $\mathbf{a} \leftarrow \mathbf{z} \rightarrow \mathbf{y}$, $\mathbf{z}$ is the common cause (confounder) of $\mathbf{a}$ and $\mathbf{x}$.

2.2 Types Of Distribution Shifts On Graph Data

Based on the proposed SCM, we redefine the covariate shift and the concept shift on graph data. Bearing in mind that OOD generalization is impossible without any assumption in the data generalization process, we make a constraint such that the conditional distribution between the latent variable $\mathbf{z}$ and the ground truth label $\mathbf{y}$ should remain invariant across different domains (e.g. $\mathcal{M}_y^{\text{train}} = \mathcal{M}_y^{\text{test}}$, $\mathbf{b}_y^{\text{train}} = \mathbf{b}_y^{\text{test}}$or $P^{\text{train}}(\mathbf{y}|\mathbf{z}) = P^{\text{test}}(\mathbf{y}|\mathbf{z})$). We justify it by pointing out that both $\mathbf{z}$ and $\mathbf{y}$ characterize the intrinsic nature of the data point, and the invariance of their correlation is necessary to ensure that OOD generalization is possible. Starting with that, we identify different sources of distribution shifts. We attribute covariate shift to the drift of the latent variable $\mathbf{z}$.

Definition 2. *(Covariate Shift):* $P^{train}(\mathbf{z}) \neq P^{test}(\mathbf{z})$ *while* $\mathcal{M}_x, \mathbf{b}_x, \mathcal{M}_s, \mathbf{b}_s, \mathcal{M}_o, \mathbf{b}_o$ *remain the same for training and test data.*

We attribute concept shift to the changes in the generation process of node features or the edges with a fixed $\mathbf{z}$ distribution. We define each case, separately.

Definition 3. *(Concept Shift-X):* $P^{train}(\mathbf{z}) = P^{test}(\mathbf{z})$ *while* $(\mathcal{M}_x^{train}, \mathbf{b}_x^{train}) \neq (\mathcal{M}_x^{test}, \mathbf{b}_x^{test})$ *and the rest parameters remain the same.*

Definition 4. *(Concept Shift-A):* $P^{train}(\mathbf{z}) = P^{test}(\mathbf{z})$ *while* $(\mathcal{M}_s^{train}, \mathbf{b}_s^{train}) \neq (\mathcal{M}_s^{test}, \mathbf{b}_s^{test})$ *and the rest parameters remain the same.*

We investigate and summarize the behavior of the joint distribution of $\mathbf{x}$ and $\mathbf{a}$ and the relations between other variables under different distribution shifts in Appendix E. As it shows, under all three types of distribution shifts, the distribution $p(\mathbf{y}|\mathbf{a}, \mathbf{x})$ changes, thus the directly estimated correlation between $\mathbf{y}$ and $\mathbf{a}, \mathbf{x}$ with the training set may fail in the test set. On the other hand, at least one of $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{y}|\mathbf{a})$ remains constant under the shifts.

2.3 Graph Decoupling

We propose to estimate $p(\mathbf{y}|\mathbf{a}, \mathbf{x})$ as the combination of $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{y}|\mathbf{a})$. However, when predicting with a graph neural network, $p(\mathbf{y}|\mathbf{a}, \mathbf{x})$ is often modeled as $\sigma(\mathbf{h}_y) = \sigma(\Psi(\mathbf{a}, \mathbf{x}))$, where $\mathbf{h}_y = \Psi(\cdot)$ represents the output embedding of the GNN model before the activation, and $\sigma(\cdot)$ is the non-linear activation function. Due to the non-linearity of $\sigma(\cdot)$, the assumption that $p(\mathbf{y}|\mathbf{a}, \mathbf{x})$ can be estimated as the combination of $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{y}|\mathbf{a})$ can be easily violated. Also, the mapping to the probability space before the combination could potentially lose the rich information delivered by the embedding space. To address this, we perform the combination process in the embedding space instead.

Assumption 2. *The value of* $\Psi(\mathbf{x}, \mathbf{a})$ *can be estimated as the weighted average of the values of individual functions* $\Psi_{\mathbf{x}}(\mathbf{x})$ *and* $\Psi_{\mathbf{a}}(\mathbf{a})$*.*

$$\Psi(\mathbf{x}, \mathbf{a}) = \gamma \cdot \Psi_{\mathbf{x}}(\mathbf{x}) + (1 - \gamma) \cdot \Psi_{\mathbf{a}}(\mathbf{a}), \tag{5}$$

where $0 \leq \gamma \leq 1$ *is a constant hyperparamter that controls the ratio between the contribution of* $\Psi_{\mathbf{x}}(\mathbf{x})$ *and* $\Psi_{\mathbf{a}}(\mathbf{a})$ *to the final prediction.*

We claim that Assumption 2 is a reasonable assumption when $\Psi(\mathbf{x}, \mathbf{a})$ is modeled with message-passing graph neural networks (MPGNNs). Most MPGNNs obtain the representation of the central node by aggregating its neighboring nodes with operations like summing or (weighted) averaging, which can be viewed as a process in which each central/neighboring node shares its "vote" on deciding the outcome. To better make our point, we analyze classification on the Simple Graph Convolution (SGC) model [Wu et al., 2019]. We show that when a fixed weight γ is assigned to the central node during the aggregation process, the prediction function of SGC can be written as:

$$\begin{aligned} \hat{y} = \sigma\Big(\gamma \underbrace{[(\tilde{D}^k)^{-\frac{1}{2}} A \tilde{A}^k (\tilde{D}^k)^{\frac{1}{2}} \mathbf{X}\Theta]_i}_{\Psi_{\mathbf{a}}(\mathbf{a}_i)} \\ + (1-\gamma) \underbrace{[((\tilde{D}^k)^{-\frac{1}{2}} I (\tilde{D}^k)^{\frac{1}{2}})^k \mathbf{X}\Theta]_i}_{\Psi_{\mathbf{x}}(\mathbf{x_i})}\Big), \end{aligned} \tag{6}$$

where $\hat{y}$ is the prediction of the model, A is the adjacency matrix, and $\tilde{A} = A + I$, $\tilde{D}$ is the degree matrix of $\tilde{A}$, k is the number of layers, Θ is the parameterized weights of each layer into a single matrix: $\Theta = \Theta_0\Theta_1...\Theta_k$. The details of the equation derivation can be found in Appendix G. We can view $[(\tilde{D}^k)^{-\frac{1}{2}} A \tilde{A}^k (\tilde{D}^k)^{\frac{1}{2}} \mathbf{X}\Theta]_i$ as $\Psi_{\mathbf{a}}(\mathbf{a}_i)$, and $[((\tilde{D}^k)^{-\frac{1}{2}} I (\tilde{D}^k)^{\frac{1}{2}})^k \mathbf{X}\Theta]_i$ as $\Psi_{\mathbf{x}}(\mathbf{x_i})$, which aligns with asssumtion 2. Notice for $\Psi_{\mathbf{a}}(\mathbf{a}_i)$ the adjacency matrix at the first layer does not include self-loops, ensuring that the node feature of the instance itself is not included.

We thus claim that Assumption 2 is reasonable. Note that it only holds when $\Psi_{\mathbf{x}}(\cdot)$ and $\Psi_{\mathbf{a}}(\cdot)$ are unbiased estimations. As shown in Figure 2 (a), $\mathbf{a}$ and $\mathbf{x}$ are caused by a common factor $\mathbf{z}$, thus they act like confounders of each other through back-door paths $\mathbf{x} \leftarrow \mathbf{z} \rightarrow \mathbf{a} \rightarrow \mathbf{y}$ and $\mathbf{a} \leftarrow \mathbf{z} \rightarrow \mathbf{x} \rightarrow \mathbf{y}$. The direct estimation of $\Psi(\mathbf{x})$ and $\Psi(\mathbf{a})$ is biased. In the next section, we aim to obtain unbiased estimations of $\Psi_{\mathbf{x}}(\cdot)$ and $\Psi_{\mathbf{a}}(\cdot)$ by considering the confounder effect.

2.4 Casual Effect Estimation

To clearly understand how changing $\mathbf{a}/\mathbf{x}$ directly influences $\mathbf{h}^y$ while considering and adjusting for other confounding factors, we propose to treat $\Psi_{\mathbf{x}}(\mathbf{x})$ and $\Psi_{\mathbf{a}}(\mathbf{a_i})$ as treatment effect of $\mathbf{x}$ and $\mathbf{a}$ on the outcome $\mathbf{h}^y$ (Section. 2.3). In a generalized causal effect estimation framework where the treatment can be a continuous representation instead of binary values, we can interpret the effect of treatment $\mathbf{t}$ as *"what additional information* $\mathbf{t}$ *can provide on predicting the outcome"*. In that sense, we want to estimate the treatment effect of $\mathbf{x}$ and $\mathbf{a}$, separately, so each can provide information about the output representation when conditioned on one another. We build two Structural Casual Models to represent the cases where one of $\mathbf{x}/\mathbf{a}$ is the treatment and another is the confounder, as shown in Figure 2 (b). Based on the SCMs, we aim to utilize casual effect inference to estimate the following Conditional Average Treatment Effects (CATEs):

$$\begin{aligned}\Psi_{\mathbf{a}}(\mathbf{a}) \triangleq & \tau(\mathbf{a}', \mathbf{a}, \mathbf{x}) \\ = & \mathbb{E}[\mathbf{h}^y | C = \mathbf{x}, do(T = \mathbf{a})] \\ & - \mathbb{E}[\mathbf{h}^y | C = \mathbf{x}, do(T = \mathbf{a}')],\end{aligned} \tag{7}$$

$$\begin{aligned}\Psi_{\mathbf{x}}(\mathbf{x}) \triangleq & \tau(\mathbf{x}', \mathbf{x}, \mathbf{a}) \\ = & \mathbb{E}[\mathbf{h}^y | C = \mathbf{a}, do(T = \mathbf{x})] \\ & - \mathbb{E}[\mathbf{h}^y | C = \mathbf{a}, do(T = \mathbf{x}')],\end{aligned} \tag{8}$$

where $\mathbf{x}'$ and $\mathbf{a}'$ are counterfactual node features and neighborhood representations. Their definitions will be discussed later. Since the counterfactual outcome is unobservable, we follow the common practice and made the assumptions in Appendix F to estimate the CATEs.

We then propose a practical solution to estimate $\tau(\mathbf{a}', \mathbf{a}, \mathbf{x})$ and $\tau(\mathbf{x}', \mathbf{x}, \mathbf{a})$, and combine them to make predictions. We apply *Generalized Robinson Decomposition (GRD)* to isolate the causal estimands and reduce the biases when estimating CATEs.

Generalized Robinson Decomposition. GRD is proposed as a generalized version of Robinson Decomposition [Robinson, 1988] to adapt graph-structured treatment. Specifically, GRD assumes that the causal effect is a product effect:

Assumption 3. *(Product Effect) We consider the following partial parameterization of* $p(y|\mathbf{c}, \mathbf{t})$*:*

$$y = g(\mathbf{c})^\top h(\mathbf{t}) + \varepsilon, \tag{9}$$

where $\mathbf{t}$*,* $\mathbf{c}$ *are the representation of the treatment and the confounder;* $g : \mathcal{C} \to \mathbb{R}^d$*,* $h : \mathcal{T} \to \mathbb{R}^d$ *and* $\mathbb{E}[\varepsilon|\mathbf{c}, \mathbf{t}] = \mathbb{E}[\varepsilon|\mathbf{c}] = 0$*, for all* $(\mathbf{c}, \mathbf{t}) \in \mathcal{C} \times \mathcal{T}$*.*

Assumption 3 has been proven to be mild and can approximate any arbitrary bounded continuous functions with a small error bound [Kaddour et al., 2021]. We define *propensity features* as $e(\mathbf{c}) \triangleq \mathbb{E}[h(\mathbf{t})|\mathbf{c}]$ and $m(\mathbf{c}) \triangleq \mathbb{E}[\mathbf{y}|\mathbf{c}] = g(\mathbf{c})^\top e(\mathbf{c})$. Following the same steps as in Robinson Decomposition, the GRD of Equation 9 is: $\mathbf{y} - m(\mathbf{c}) = g(\mathbf{c})^\top (h(\mathbf{t}) - e(\mathbf{c})) + \varepsilon$. Given nuisance estimates $\hat{m}(\cdot)$ and $\hat{e}(\cdot)$, $g(\cdot)$ and $h(\cdot)$ can be derived with the optimization problem:

$$\begin{aligned}&\hat{g}(\cdot), \hat{h}(\cdot) \triangleq \arg\min_{g,h} \\ &\left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{y}_i - \hat{m}(\mathbf{c}_i) - g(\mathbf{c}_i)^\top \left(h(\mathbf{t}_i) - \hat{e}(\mathbf{c}_i) \right) \right)^2 \right\}\end{aligned} \tag{10}$$

With estimated $g(\cdot)$ and $h(\cdot)$, the CATE of treatment variable $\mathbf{t}$ and its counterfactual $\mathbf{t}'$ given the confounder $\mathbf{c}$ can be simplified as:

$$\tau(\mathbf{t}', \mathbf{t}, \mathbf{c}) = g(\mathbf{c})^\top \left(h(\mathbf{t}') - h(\mathbf{t}) \right). \tag{11}$$

To optimize Equation 10, we can use *Structured Intervention Networks* (SIN) [Kaddour et al., 2021], a two-stage training algorithm to learn $g(\cdot)$ and $h(\cdot)$ as neural networks and estimate the CATE with Equation 11. In our case, we need to estimate separate set decomposition functions, $g(\cdot)$ and $h(\cdot)$, for each casual model, and to apply SIN directly would bring the following drawbacks: 1) By separately applying SIN to our two casual models, it would fail to share the representations of the common factors (e.g. the confounder of one model is the treatment of another); the lack of knowledge sharing could make the learning process less efficient and the learned model prone to overfitting. 2) Unlikely common scenarios where counterfactual treatments are well-defined, in our case, $\mathbf{f}'$ and $\mathbf{a}'$ can not be easily decided. Since they are both continuous values that can span the space, *how to define their values in the situation where the node is* ***not*** *treated by the treatment (node features/neighborhood representations)?* To address these two problems, we propose *Dual Casual Decomposition* and *Background Counterfactual Selection* as components of our method.

2.4.1 Dual Casual Decomposition

We aim to learn two sets of $g(\cdot), h(\cdot)$ for SCM-X and SCM-A. We denote them as $g^X(\cdot), h^X(\cdot)$ and $g^A(\cdot), h^A(\cdot)$, separately. Each casual model is also associated with a set of $e(\cdot), m(\cdot)$, denoted as $e^X(\cdot), m^X(\cdot)$ and $e^A(\cdot), m^A(\cdot)$. Observing that directly applying SIN separately could double the training time and be inefficient, we notice that for our two SCMs, the treatment of one model is the confounder of the other. Leveraging this fact, we allow the embedding

of the same entity to be shared across the two models. We propose a new paradigm, named Dual Causal Decomposition, which first learns the common embedding of the two models and then applies a lighter dual version of SIN to estimate the remaining embedding. Our approach effectively reduces model parameters. The complete process is provided in Appendix H.

2.4.2 Background Counterfactual Selection

We view the treatment effect as useful information provided by the treatment variables for decision-making in predictions. At each of the casual models, the factual treatment ($\mathbf{a}$ or $\mathbf{x}$) received by an instance reveals information about its label. In the counterfactual "*untreated*" case, the treatment representation should reveal no such information. It is problematic to simply set the counterfactual representation as zeros since an all-zeros embedding does not necessarily mean the absence of information. Instead, for each instance, at each time, we randomly sample a treatment representation from the whole dataset, and we answer the question: *what net effect does the factual treatment bring compared to the random counterfactual treatment?* We repeat the above process multiple times and average the net effects as the estimated treatment effect.

Specifically, for SCM-A, we randomly sample k neighborhood representations with indexes $s_1, ..., s_k$ from the datasets as the counterfactual treatment. The counterfactual outcome is then estimated as follows:

$$\mathbf{E}[\mathbf{h}^y|C=\mathbf{a}, do(T=\mathbf{x}')] \triangleq \frac{1}{k}\sum_{i=1}^{k} g^A(a_{s_i})^\top h^A(f_{s_i}). \tag{12}$$

Similarly, for SCM-X, the counterfactual outcome is estimated as:

$$\mathbf{E}[\mathbf{h}^y|C=\mathbf{x}, do(T=\mathbf{a}')] \triangleq \frac{1}{k}\sum_{i=1}^{k} g^X(f_{s_i})^\top h^X(a_{s_i}). \tag{13}$$

We then estimate $\Psi_{\mathbf{a}}(\mathbf{a})$ and $\Psi_{\mathbf{x}}(\mathbf{x})$ with Equation 7 and 8 and make a prediction:

$$\hat{\mathbf{y}} = \sigma\big(\gamma \cdot \Psi_{\mathbf{x}}(\mathbf{x}) + (1-\gamma)\cdot \Psi_{\mathbf{a}}(\mathbf{a})\big). \tag{14}$$

We provide the pseudo-code for **DeCaf** in Algorithm 1, and a summary of important notations in Table 5.

3 EXPERIMENTS

To evaluate the effectiveness of **DeCaf**, we aim to answer the following research questions (**RQs**): **RQ1:** How well can **DeCaf** handle *covariate shift*? **RQ2:** How well can **DeCaf** handle *concept shift*? **RQ3:** How does the confounder effect between node feature $\mathbf{x}$ and the neighborhood representation $\mathbf{a}$ impact the performance of different models, and how well can **DeCaf** handle this confounder effect?

3.1 Comparison Methods

We compare **DeCaf** with SOTA Graph OOD generalization methods that are applicable to the node classification including IRM [Arjovsky et al., 2020], REX [Krueger et al., 2020], EERM [Wu et al., 2022a], CIT [Xia et al., 2023], FLOOD [Liu et al., 2023], and StableGL [Zhang et al., 2023]. We also compare with empirical risk minimization (ERM) as a baseline. Among these methods, REX [Krueger et al., 2020] requires access to multiple training environments, thus it is only applicable to the `OGB-elliptic` and `Facebook-100` datasets with multiple training graphs. SR-GNN [Zhu et al., 2021] requires access to the input distribution of the test set when training the model and does not apply to the inductive setting studied in this paper. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU with 24GB memory. We compare **DeCaf** with the baseline methods regarding their restrictions and complexity in Appendix J.

Table 1: Test Macro-F1 scores on single-graph datasets with soft label-leaveout. "OOM" stands for out of memory. The best results are bold-faced.

Dataset	Method	SGC	GCN	GAT
Cora	ERM	65.61±4.28	67.94±0.89	66.95±2.81
	IRM	65.52±3.01	66.04±2.33	65.46±4.08
	EERM	65.27±2.22	67.24±2.86	69.50±2.66
	CIT	60.51±1.48	61.30±0.72	67.80±2.28
	FLOOD	63.58±4.86	62.26±5.54	66.35±5.43
	StableGL	59.96±13.18	65.42±5.20	67.28±1.11
	DeCaf	**71.41±1.19**	**70.12±1.29**	**70.58±0.49**
Citeseer	ERM	49.05±3.15	49.58±1.47	52.90±0.58
	IRM	52.98±1.86	51.82±1.59	51.75±1.00
	EERM	44.53±0.92	43.56±2.10	52.63±4.78
	CIT	44.21±5.75	49.36±0.64	55.94±1.94
	FLOOD	46.75±4.78	49.56±5.49	53.08±0.51
	StableGL	51.35±7.34	49.62±3.29	51.50±1.12
	DeCaf	**59.77±1.15**	**58.85±0.51**	**56.71±1.96**
Amazon	ERM	86.67±1.11	87.14±0.33	87.71±0.95
	IRM	86.96±0.14	88.01±0.40	86.79±1.05
	EERM	87.63±0.50	88.12±0.16	86.68±1.43
	CIT	87.83±0.92	87.46±0.91	82.28±4.51
	FLOOD	87.08±0.73	87.33±0.68	85.20±3.72
	StableGL	87.26±0.73	87.48±0.45	85.26±2.82
	DeCaf	**89.67±0.39**	**88.93±0.73**	**88.74±0.57**
Coauthor	ERM	86.60±0.91	87.66±0.24	80.48±1.21
	IRM	87.96±0.52	88.75±0.58	78.87±1.20
	EERM	84.68±1.28	85.27±1.04	OOM
	CIT	84.48±0.40	86.17±1.52	**85.60±0.84**
	FLOOD	83.63±1.13	85.27±1.04	OOM
	StableGL	84.48±0.40	86.17±1.52	85.93±0.76
	DeCaf	**89.20±0.37**	**88.97±0.26**	85.28±1.42

3.2 Performance On Covariate Shift (RQ1)

We use seven single-graph real-world datasets in which `Cora`, `Citeseer`, `Amazon-photo` and `Coauthor-CS` are homophilous graphs; `Squirrel`, `Roman-empire` and `Tolokers` are heterophilous graphs. Further details of the above datasets are provided in Appendix K.3.

As the above datasets have no clear domain information, we synthetically create OOD data with *soft label-leaveout*, which is inspired by *label-leaveout* used in OOD detection [Wu et al., 2023]. For OOD generalization, the model is not expected to predict unseen classes, so instead of completely leaving out partial classes to the test set, we allow the training set to have a small portion of samples from those classes. In our experiments, we make sure that the training, validation, and test sets have different class distributions. By splitting the samples into groups with different class distributions but a relatively constant *input-label* relationship, *Soft Label-Leaveout* simulates covariate shift based on Definition 2.

Table 2: Test F1 scores on heterophilous graphs with soft label-leaveout with H2GCN as backbone.

	Squirrel	Roman-empire	Tolokers
ERM	24.12±3.52	42.01±0.98	46.94±3.07
IRM	28.81±2.06	40.58±0.60	47.86±2.26
EERM	30.42±3.71	OOM	44.18±0.18
CIT	28.93±1.80	45.41±3.25	44.26±0.00
FLOOD	23.96±8.73	OOM	44.58±0.18
StableGL	26.44±4.73	45.78±2.09	44.10±0.20
DeCaf	**32.57±1.70**	**48.85±0.70**	**60.14±0.51**

Table 3: Test F1 scores on Facebook-100 using GNN with the best validation F1.

Training	Johns Hopkins + Caltech + Amherst		
Test	Penn	Brown	Texas
ERM	49.23±1.72	49.68±0.93	48.57±0.21
IRM	35.26±2.40	46.92±5.66	36.86±1.64
REX	44.77±6.48	42.65±7.34	44.05±8.88
EERM	22.62±22.91	49.44±1.92	49.12±1.71
CIT	44.66±6.65	45.26±6.21	42.10±8.97
FLOOD	42.37±5.06	41.48±5.28	40.82±5.94
StableGL	44.54±6.58	45.64±6.25	43.75±5.65
CaNet	48.73±1.06	50.42±0.61	48.92±0.80
DeCaf	**55.31±0.40**	**53.31±0.11**	**53.56±0.19**

We compare **DeCaf** with the baselines on the four homophilous graph datasets with soft label-leaveouts, `Cora`, `Citeseer`, `Amazon-photo`, and `Coauthor-CS`, with SGC, GCN, and GAT as GNN backbones. We report the Macro-F1 score in Table 1. We observe that **DeCaf** outperforms ERM across all cases with an average improvement of 4.2%, demonstrating its ability to mitigate the negative impact of covariate shifts. Also, **DeCaf** beats the best baselines in most cases with an average improvement of 2.4%, showing that it better handles covariate shifts.

We compare **DeCaf** with the baselines on three heterophilous graph datasets with *soft label-leaveouts*, `Squirrel`, `Roman-empire`, and `Tolokers`. Instead of using common GNNs that are not suited for heterophilous graphs [Platonov et al., 2024], we use H2GCN [Zhu et al., 2020], a SOTA GNN well established for graph heterophily problems, as the backbone

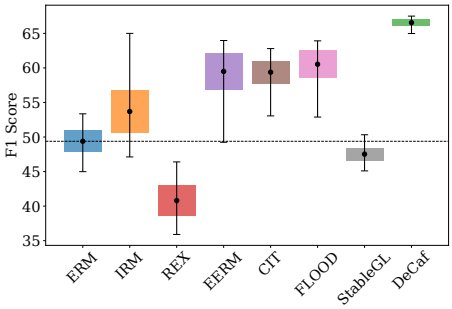


Figure 3: Distribution of F1 scores on `OGB-elliptic` of different models shown in a bar plot. The dashed line shows the mean F1 score of the ERM method.

GNN for these datasets. We report the performance of **DeCaf** compared to the baselines in Table 2. We report Micro F1 score for `Squirrel`, `Roman-empire` with multiple classes, and F1 score for `Tolokers` with binary classes. On average, **DeCaf** improves the (Macro-) F1 scores by 5.8%. We observe that **DeCaf** outperforms the baselines among all datasets.

3.3 Performance On Concept Shift (RQ2)

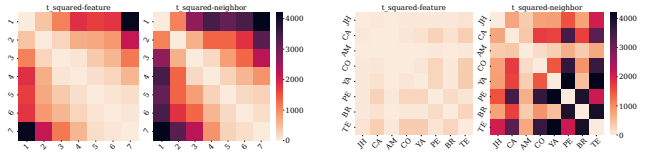
`Facebook-100` [Traud et al., 2012] contains 100 social networks collected from universities in the United States. Following [Wu et al., 2022a], we adapt three graphs for training, two graphs for validation, and three graphs for testing. Three sets of training graphs are used. `OGB-elliptic` [Rozemberczki et al., 2021] is a dynamic financial network dataset that contains 43 graph snapshots from different time steps. We use the first 5 graph snapshots for training, the next 5 for validation, and the last 33 for testing. Further details of the two datasets are provided in Appendix K.3.

We first identify distribution shifts within both datasets. To assess concept shifts, we employ Hotelling's T-squared statistics to measure the disparity between node distributions sharing the same class label. Our intuition is that if the nodes from the same class in two graphs have significantly different distributions, we can claim the graphs exhibit a concept shift with different *feature/structure-label* mappings. We present the pairwise T-squared scores comparing node feature embeddings and neighborhood representations of nodes from the first class across subgraphs for `Facebook-100` (Figure 4 (a)) and `OGB-elliptic` (Figure 4 (a)). A higher T-squared score indicates a greater dissimilarity between the two distributions. In Figure 4, we observe that the subgraphs within `Facebook-100` experience significantly less extent shift in node features compared to neighborhood representations, suggesting a dominance of neighborhood shift (def. concept shift-A). For `OGB-elliptic`, we initially

Table 4: Test Macro-F1 scores on synthetic datasets. The best results are bold-faced.

		h-feat	qrt-feat	full-feat
best GNN	ERM	40.78±11.45	21.68±3.81	41.76±0.63
	IRM	38.10±2.54	12.86±4.47	13.16±6.42
	EERM	35.15±15.09	39.46±10.48	**54.71±0.74**
	CIT	30.83±25.93	18.37±11.45	38.28±1.70
	FLOOD	47.22±4.16	35.59±21.58	42.17±9.55
	StableGL	32.83±12.24	28.56±20.37	45.12±0.77
	DeCaf	**57.24±13.75**	**49.00±2.61**	54.11±1.14
H2GCN	ERM	49.38±3.44	31.72±1.60	66.59±1.86
	IRM	51.17±8.82	33.57±4.78	62.04±1.45
	EERM	49.04±3.54	30.04±1.72	65.67±1.62
	CIT	54.33±3.90	30.28±3.96	64.78±2.64
	FLOOD	47.18±4.97	29.56±1.13	64.65±0.79
	StableGL	39.99±2.97	37.44±5.21	63.06±4.10
	DeCaf	**55.92±5.20**	**44.96±1.78**	**67.02±1.12**

notice a gradual increase in the T-squared score across the axis, indicating that the distribution shift intensifies over time. Additionally, we observe that it experiences concept shift-A and concept shift-X more evenly, with the latter slightly more dominant.



(a) `OGB-elliptic` (b) `Facebook-100`

Figure 4: Hotelling's two-sample t-squared statistic of node feature embedding between subgraphs from different time periods in `OGB-elliptic` dataset.

For `facebook-100`, we report the F1-score of **DeCaf** in comparison with the baselines on each of the test graphs in Table 3. For each method, we use SGC, GCN, GAT, and H2GCN as baselines, and only report the results of the best-performed GNN selected with the validation set. We provide the results when using the first set of training graphs. Complete results are provided in Appendix I. We observe that **DeCaf** significantly outperforms best baselines with an average improvement of 3.8%, showing that **DeCaf** enhances the generalizability of GNNs across different domains.

For `OGB-elliptic`, we plot the distribution of the F1 score averaged on all of the test graph snapshots with SGC as the backbone in Figure 3. The rectangles represent the standard deviation, and the error bars illustrate the range of data samples. The dashed line shows the mean F1 score of the ERM method. As shown, **DeCaf** significantly outperforms all baselines with a higher mean F1 score while demonstrating more stable performance, indicated by a smaller standard deviation, showing its ability to handle potentially more complex temporal shifts.

We also note that unlike other datasets, where in most cases the baseline graph OOD methods the baseline graph OOD methods either slightly improve or achieve comparable performance as ERM, on `facebook-100` and `OGB-elliptic`, most of them degrade the performance significantly. The reasons behind this could be that the false assumptions made by those methods fail to apply to the distribution shifts in real-life scenarios and the over-confidence brought by their strategies could further degrade the generalizability of the model.

3.4 The Impact Of Confounder Effect (RQ3)

We create three synthetic graphs, `h-feat`, `qtr-feat`, and `full-feat`, to simulate scenarios with and without confounder effects between node features and neighborhood representations. `h-feat` is created such that the node features and neighborhood representations are dependent and act as confounders to each other. In contrast, `full-feat` and `qtr-feat` are designed so that these elements are independent, with no confounder effects. Details for creating these datasets are provided in Appendix K.4.

We compare our model against baselines on the three synthetic datasets. We report the results of the best-performed one between SGC, GCN, GAT selected with the validation set. We also incorporate H2GCN to address potential heterophily in these datasets. Besides, H2GCN separately models node features and neighborhood representations, making its performance on *ERM* a useful benchmark for our decoupling framework. We present the average Macro-F1 scores in Table 4. We report the results for the best-performed one among SGC, GCN, and GAT due to page limitation, please refer to Table 7 in Appendix for the full results. For `qtr-feat` and `full-feat`, where node features and neighborhood representations are independent, employing H2GCN on ERM significantly improves the Macro-F1 score compared to other GNN models. However, in `h-feat`, where node features and neighborhood representations are correlated, the gains are modest and are outperformed by our model with any GNN backbone. This suggests that while the separate modeling approach of H2GCN can mitigate distribution shifts, it falls short in addressing confounder effects, thus underperforming when node features and neighborhood representations are correlated.

4 RELATED WORK

Prior works [Ahuja et al., 2020, Bai et al., 2021, Sagawa et al., 2019, Shen et al., 2021, Ye et al., 2021] have been dedicated to addressing the out-of-distribution (OOD) generalization problem. The primary objective of OOD generalization is to train models within a specific domain and anticipate

their robust generalization to test domains from potentially distinct distributions [Ye et al., 2021]. While most of these methods are tailored to deal with distribution shifts on tabular data or images, their performance is restricted when confronted with more complex data structures. With the success of Graph Neural Networks (GNNs) [Veličković et al., 2018, Kipf and Welling, 2016], recent research starts to address graph OOD [Li et al., 2023, Fan et al., 2024, Sui et al., 2021, Wu et al., 2022c, Chen et al., 2022, Li et al., 2022b, Wu et al., 2022b, Buffelli et al., 2022, Fan et al., 2022, Wu et al., 2022a, Liu et al., 2023, Xia et al., 2023, Zhang et al., 2023, Zhu et al., 2021]. OOD generalization on graphs is more intractable due to the *non-euclidean* nature of graph data structures [Qiao et al., 2024, Ai et al., 2025, Li et al., 2024, Qiao and Pang, 2024], and the subtlety of different types of distribution shifts [Li et al., 2022a].

One consensus is that, when domain knowledge is unavailable, knowledge transfer to a new domain is impossible without structural assumption on data generation processes [Wu et al., 2022a]. A general assumption made by most exisitng methods is that there exist "true" correlations between input data features and their labels. These correlations remain invariant across different domains, and they aim to identify these true connections while removing the spurious ones [Li et al., 2022a]. These methods leverage this assumption in different manners or extend it to more specific forms. Approaches [Li et al., 2023, Fan et al., 2024, Fan et al., 2022] backed by *confounder balancing* [Kuang et al., 2018] from casual theories remove the correlations between casual and non-causal (spurious) aspects (in the forms of representations [Li et al., 2023], subgraphs [Fan et al., 2024], or samples [Fan et al., 2022], etc.) such that the model can focus on the casual ones. Some methods [Li et al., 2022b, Wu et al., 2022b, Wu et al., 2022a, Liu et al., 2023, Xia et al., 2023, Zhang et al., 2023] leverage *invariance principle* [Arjovsky et al., 2020] from causality and assumes that there exists a portion of information in the input (e.g., subgraphs) that is invariant to label predictions across different environments. This approach often requires access to multiple environments/domains during the training process, which are not always available. Structural causal graphs (SCGs) have also been used for assumptions in data generation processes [Sui et al., 2021, Wu et al., 2022c, Chen et al., 2022]. They focus on making unbiased estimations of causal relationships via *do*-calculus [Sui et al., 2021] or back-door adjustments [Wu et al., 2022c]. [Zhu et al., 2021] assume that training data are from a biased data generation process while test data are unbiased. [Buffelli et al., 2022] improves model generalization on smaller or larger test graphs by minimizing the discrepancy between the learned representation on the original graph and the coarsened graph.

Among the above-mentioned methods, most of them [Li et al., 2023, Fan et al., 2024, Sui et al., 2021, Wu et al., 2022c, Chen et al., 2022, Li et al., 2022b, Wu et al., 2022b, Buffelli et al., 2022] are tailored for graph-level or link-level tasks, and it is non-trivial to adapt them to node-level tasks. For the rest of the methods [Fan et al., 2022, Wu et al., 2022a, Liu et al., 2023, Xia et al., 2023, Zhang et al., 2023, Zhu et al., 2021] that can be applied on node-level tasks, an ego-net is often the starting point for any further actions to be taken (to generate an invariant subgraph, or to learn the invariant representation of it, etc). Current approaches involve aggregating a central node and its neighborhood into a unified representation (e.g., GCNs [Kipf and Welling, 2016] or GATs [Veličković et al., 2018]), but they tend to overlook and filter out potential complex dependencies (or independencies) between them. Certain GNNs such as H2GCN [Zhu et al., 2020] can model the central node and its neighbors' representations separately through the concatenation operation during the aggregation, but inferences on their relations are still missing. Disentangled graph learning [Ma et al., 2019, Liu et al., 2020] investigates latent factors that may cause the formation of an edge between a node and its neighbors. This method can also improve the OOD generalizability of GNNs, but it focuses on explaining the existence of edges with node representations, which is fundamentally different from our focus.

5 CONCLUSION

In this paper, we introduce a causal decoupling framework to improve out-of-distribution generalization on node classification. We develop a comprehensive theoretical foundation to show that, by independently estimating the treatment effects of node features and neighborhood representations, the casual decoupling framework is robust when dealing with different types of distribution shifts. Following this theoretical basis, we also propose an implementable end-to-end framework, **DeCaf**, that leverages casual inference technologies to estimate the decoupled representations and make final predictions. We demonstrate the effectiveness and the power of **DeCaf** for node classification on both real-world datasets and synthetic datasets under different types of distribution shifts.

Acknowledgement

We thank the anonymous reviewers for their helpful comments. This work is supported in part by the US National Science Foundation under grants 2047843 and 2437621.

References

[Abadie and Imbens, 2006] Abadie, A. and Imbens, G. W. (2006). Large sample properties of matching estimators for average treatment effects. *Econometrica*, 74(1):235–267.

[Ahuja et al., 2020] Ahuja, K., Shanmugam, K., Varshney, K. R., and Dhurandhar, A. (2020). Invariant risk minimization games. *CoRR*, abs/2002.04692.

[Ai et al., 2025] Ai, G., Pang, G., Qiao, H., Gao, Y., and Yan, H. (2025). Grokformer: Graph fourier kolmogorov-arnold transformers.

[Arjovsky et al., 2020] Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2020). Invariant risk minimization.

[Athey and Wager, 2019] Athey, S. and Wager, S. (2019). Estimating treatment effects with causal forests: An application. *Observational Studies*, 5.

[Austin, 2011] Austin, P. C. (2011). An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behavioral Research*, 46(3):399–424. PMID: 21818162.

[Bai et al., 2021] Bai, H., Sun, R., Hong, L., Zhou, F., Ye, N., Ye, H.-J., Chan, S.-H. G., and Li, Z. (2021). Decaug: Out-of-distribution generalization via decomposed feature representation and semantic augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6705–6713.

[Bica et al., 2020] Bica, I., Jordon, J., and van der Schaar, M. (2020). Estimating the effects of continuous-valued interventions using generative adversarial networks. *CoRR*, abs/2002.12326.

[Buffelli et al., 2022] Buffelli, D., Lio, P., and Vandin, F. (2022). Sizeshiftreg: a regularization method for improving size-generalization in graph neural networks. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

[Chang and Dy, 2017] Chang, Y. and Dy, J. (2017). Informative subspace learning for counterfactual inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

[Chen et al., 2022] Chen, Y., Zhang, Y., Bian, Y., Yang, H., KAILI, M., Xie, B., Liu, T., Han, B., and Cheng, J. (2022). Learning causally invariant representations for out-of-distribution generalization on graphs. In *Advances in Neural Information Processing Systems*.

[Chernozhukov et al., 2017] Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., and Robins, J. (2017). Double/debiased machine learning for treatment and causal parameters.

[Fan et al., 2024] Fan, S., Wang, X., Shi, C., Cui, P., and Wang, B. (2024). Generalizing graph neural networks on out-of-distribution graphs. *IEEE Transactions on pattern analysis and machine intelligence*, 46.

[Fan et al., 2022] Fan, S., Wang, X., Shi, C., Kuang, K., Liu, N., and Wang, B. (2022). Debiased graph neural networks with agnostic label selection bias. *CoRR*, abs/2201.07708.

[Funk et al., 2011] Funk, M. J., Westreich, D., Wiesen, C., Stürmer, T., Brookhart, M. A., and Davidian, M. (2011). Doubly Robust Estimation of Causal Effects. *American Journal of Epidemiology*, 173(7):761–767.

[Giles et al., 1998] Giles, C. L., Bollacker, K. D., and Lawrence, S. (1998). Citeseer: an automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, page 89–98, New York, NY, USA. Association for Computing Machinery.

[Gui et al., 2022] Gui, S., Li, X., Wang, L., and Ji, S. (2022). Good: A graph out-of-distribution benchmark.

[Harada and Kashima, 2020] Harada, S. and Kashima, H. (2020). Graphite: Estimating individual effects of graph-structured treatments. *CoRR*, abs/2009.14061.

[Heckman et al., 2018] Heckman, J. J., Humphries, J. E., and Veramendi, G. (2018). Returns to education: The causal effects of education on earnings, health, and smoking. *Journal of Political Economy*, 126(S1):S197–S246.

[Hill, 2011] Hill, J. L. (2011). Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240.

[Johansson et al., 2016] Johansson, F., Shalit, U., and Sontag, D. (2016). Learning representations for counterfactual inference. In *Proceedings of The*

33rd International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, pages 3020–3029, New York, New York, USA. PMLR.

[Kaddour et al., 2021] Kaddour, J., Zhu, Y., Liu, Q., Kusner, M. J., and Silva, R. (2021). Causal effect inference for structured treatments. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24841–24854. Curran Associates, Inc.

[Kallus, 2020] Kallus, N. (2020). DeepMatch: Balancing deep covariate representations for causal inference using adversarial training. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5067–5077. PMLR.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

[Krueger et al., 2020] Krueger, D., Caballero, E., Jacobsen, J., Zhang, A., Binas, J., Priol, R. L., and Courville, A. C. (2020). Out-of-distribution generalization via risk extrapolation (rex). *CoRR*, abs/2003.00688.

[Kuang et al., 2018] Kuang, K., Xiong, R., Cui, P., Athey, S., and Li, B. (2018). Stable prediction across unknown environments. *CoRR*, abs/1806.06270.

[Künzel et al., 2019] Künzel, S. R., Sekhon, J. S., Bickel, P. J., and Yu, B. (2019). Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165.

[Li et al., 2024] Li, H., Fu, J., Ling, X., Sun, Z., Wang, K., and Chen, Z. (2024). Single-cell curriculum learning-based deep graph embedding clustering.

[Li et al., 2022a] Li, H., Wang, X., Zhang, Z., and Zhu, W. (2022a). Out-of-distribution generalization on graphs: A survey.

[Li et al., 2023] Li, H., Wang, X., Zhang, Z., and Zhu, W. (2023). Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):7328–7340.

[Li et al., 2022b] Li, H., Zhang, Z., Wang, X., and Zhu, W. (2022b). Learning invariant graph representations for out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, volume 35, pages 11828–11841. Curran Associates, Inc.

[Liu et al., 2023] Liu, Y., Ao, X., Feng, F., Ma, Y., Li, K., Chua, T.-S., and He, Q. (2023). Flood: A flexible invariant learning framework for out-of-distribution generalization on graphs. KDD '23, page 1548–1558, New York, NY, USA. Association for Computing Machinery.

[Liu et al., 2020] Liu, Y., Wang, X., Wu, S., and Xiao, Z. (2020). Independence promoted graph disentangled networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4916–4923.

[Ma et al., 2019] Ma, J., Cui, P., Kuang, K., Wang, X., and Zhu, W. (2019). Disentangled graph convolutional networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4212–4221. PMLR.

[McAuley et al., 2015] McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes.

[McCallum et al., 2000] McCallum, A., Nigam, K., Rennie, J. D. M., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

[Platonov et al., 2024] Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. (2024). A critical look at the evaluation of gnns under heterophily: Are we really making progress?

[Prosperi et al., 2020] Prosperi, M. C. F., Guo, Y., Sperrin, M., Koopman, J. S., Min, J., He, X., Rich, S. N., Wang, M., Buchan, I. E., and Bian, J. (2020). Causal inference and counterfactual prediction in machine learning for actionable healthcare. *Nature Machine Intelligence*, 2:369 – 375.

[Qiao and Pang, 2024] Qiao, H. and Pang, G. (2024). Truncated affinity maximization: One-class homophily modeling for graph anomaly detection.

[Qiao et al., 2024] Qiao, H., Wen, Q., Li, X., Lim, E.-P., and Pang, G. (2024). Generative semi-supervised graph anomaly detection.

[Robinson, 1988] Robinson, P. M. (1988). Root-n-consistent semiparametric regression. *Econometrica*, 56(4):931–954.

[Rozemberczki et al., 2021] Rozemberczki, B., Allen, C., and Sarkar, R. (2021). Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014.

[Rubin, 1973] Rubin, D. B. (1973). Matching to remove bias in observational studies. *Biometrics*, 29(1):159–183.

[Rubin, 1978] Rubin, D. B. (1978). Using multivariate matched sampling and regression adjustment to control bias in observational studies. *ETS Research Bulletin Series*, 1978(2):i–33.

[Sagawa et al., 2019] Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. (2019). Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR*, abs/1911.08731.

[Shalit et al., 2017] Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms.

[Shchur et al., 2019] Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2019). Pitfalls of graph neural network evaluation.

[Shen et al., 2021] Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. (2021). Towards out-of-distribution generalization: A survey. *CoRR*, abs/2108.13624.

[Shi et al., 2019] Shi, C., Blei, D. M., and Veitch, V. (2019). Adapting neural networks for the estimation of treatment effects.

[Sui et al., 2021] Sui, Y., Wang, X., Wu, J., He, X., and Chua, T. (2021). Deconfounded training for graph neural networks. *CoRR*, abs/2112.15089.

[Traud et al., 2012] Traud, A. L., Mucha, P. J., and Porter, M. A. (2012). Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180.

[Veličković et al., 2018] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

[Wager and Athey, 2018] Wager, S. and Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242.

[Wu et al., 2019] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

[Wu et al., 2023] Wu, Q., Chen, Y., Yang, C., and Yan, J. (2023). Energy-based out-of-distribution detection for graph neural networks. In *The Eleventh International Conference on Learning Representations*.

[Wu et al., 2022a] Wu, Q., Zhang, H., Yan, J., and Wipf, D. (2022a). Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*.

[Wu et al., 2022b] Wu, Y., Wang, X., Zhang, A., He, X., and Chua, T. (2022b). Discovering invariant rationales for graph neural networks. *CoRR*, abs/2201.12872.

[Wu et al., 2022c] Wu, Y.-X., Wang, X., Zhang, A., Hu, X., Feng, F., He, X., and Chua, T.-S. (2022c). Deconfounding to explanation evaluation in graph neural networks.

[Wu et al., 2021] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

[Xia et al., 2023] Xia, D., Wang, X., Liu, N., and Shi, C. (2023). Learning invariant representations of graph neural networks via cluster generalization. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[Yao et al., 2018] Yao, L., Li, S., Li, Y., Huai, M., Gao, J., and Zhang, A. (2018). Representation learning for treatment effect estimation from observational data. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Ye et al., 2021] Ye, H., Xie, C., Cai, T., Li, R., Li, Z., and Wang, L. (2021). Towards a theoretical framework of out-of-distribution generalization.

[Zhang et al., 2023] Zhang, S., Tong, Y., Kuang, K., Feng, F., Qiu, J., Yu, J., Zhao, Z., Yang, H., Zhang, Z., and Wu, F. (2023). Stable prediction on graphs with agnostic distribution shifts. In *Proceedings of The KDD'23 Workshop on Causal Discovery, Prediction and Decision*, volume 218 of *Proceedings of Machine Learning Research*, pages 49–74. PMLR.

[Zhu et al., 2020] Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. (2020). Generalizing graph neural networks beyond homophily. *CoRR*, abs/2006.11468.

[Zhu et al., 2021] Zhu, Q., Ponomareva, N., Han, J., and Perozzi, B. (2021). Shift-robust GNNs: Overcoming the limitations of localized graph training

data. *Advances in Neural Information Processing Systems*, 34.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model.
   [Yes] We provide a clear description of the mathematical setting, assumptions, and the detailed methodology for the Casual Decoupling Framework we propose in section 2. We also provide the pseudocode for the main algorithm in Appendix B.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm.
   [Yes] We conduct complexity analysis in Appendix J.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries.
   [No] We will open-source our source code upon publication.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results.
   [Yes] We provide the full set of assumptions for deriving the theoretic framework, including Assumption 1, Assumption 2, and complete assumptions for CATE estimation as detailed in Appendix F.
   (b) Complete proofs of all theoretical results.
   [Yes] We provide complete proof of all theoretical results in section 2 of the main paper and in Appendix E and G.
   (c) Clear explanations of any assumptions.
   [Yes] We provide clear explanations for every assumption we make.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL).
   [No] We will open-source our source code upon publication.
   (b) All the training details (*e.g.*, data splits, hyperparameters, how they were chosen).
   [Yes] We provide all necessary training details to reproduce the reported results in Appendix K.
   (c) A clear definition of the specific measure or statistics and error bars (*e.g.*, with respect to the random seed after running experiments multiple times).
   [Yes] We provide standard deviations for all results over multiple runs. We indicate that in the experiment section of the paper.
   (d) A description of the computing infrastructure used. (*e.g.*, type of GPUs, internal cluster, or cloud provider).
   [Yes] We provide details of computations resources in Section 3.1 and Appendix K.

4. If you are using existing assets (*e.g.*, code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets.
   [Yes] We have properly cited all the assets including code, data, and models.
   (b) The license information of the assets, if applicable. [NA]
   (c) New assets either in the supplemental material or as a URL, if applicable. [NA]
   (d) Information about consent from data providers/curators. [NA]
   (e) Discussion of sensible content if applicable, *e.g.*, personally identifiable information or offensive content. [NA]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [NA]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [NA]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [NA]

DeCaf: A Causal Decoupling Framework for OOD Generalization on Node Classification: Supplementary Materials

A RELATED WORK ON CAUSAL EFFECT ESTIMATION

Estimating the causal effect of treatment plays a crucial role in many domains [Heckman et al., 2018, Prosperi et al., 2020]. With the presence of the confounder, two types of studies, dubbed *randomized controlled trials (RCTs)* and *observational studies*, are often conducted to achieve an unbiased estimation of casual effects [Yao et al., 2018]. Often, RCTs are expensive, unethical, or infeasible [Yao et al., 2018], leaving observational studies the only option. The challenges of observational studies are that the counterfactual samples are missing from observational data, and the estimation of counterfactual output is often biased due to the treatment selection bias caused by the confounder effect [Chernozhukov et al., 2017]. Traditional solutions include matching methods [Abadie and Imbens, 2006, Austin, 2011, Rubin, 1978, Chang and Dy, 2017, Rubin, 1973], tree-based methods [Hill, 2011, Wager and Athey, 2018], and regression-based methods [Künzel et al., 2019, Funk et al., 2011]. Recently, representation learning methods [Shalit et al., 2017, Yao et al., 2018, Johansson et al., 2016, Shi et al., 2019, Athey and Wager, 2019, Bica et al., 2020, Kallus, 2020] have also been widely studied and demonstrate impressive performance. The above-mentioned studies focus on binary or categorical treatments, which are difficult to apply when the treatments are real-valued and structured. GraphITE [Harada and Kashima, 2020] is proposed to deal with graph-structured treatment, which mitigates observation biases by reinforcing the independence between the treatment and covariate representations in the notion of Hilbert-Schmidt Independence Criterion (HSIC). Robinson decomposition [Kaddour et al., 2021] is used to identify the distinct contribution of the treatment and the covariates and generalizes it such that the treatment can be vectorized to a continuous embedding.

B PSEUDO CODE

We provide the pseudo-code for **DeCaf** in Algorithm 1.

C IMPORTANT MATHEMATICAL NOTATIONS

We provide a summary of important mathematical notations in Table 5.

D DISCUSSION ON SPILLOVER EFFECT

Assumption 4. *The Law of Large Numbers is invoked, ensuring that the sample size is sufficiently large such that the observed distribution of the random variable z remains stable and converges towards its theoretical distribution.*

Observation 1. *Given $\{\mathbf{z}_j\}_{j=1}^n$ are sampled from a specific distribution. Under assumption 1, the expectation of the neighborhood representation $\mathbf{a}_i$ of node i is dependent on $\mathbf{z}_i$ and the distribution of z. Under assumption 4, the change of other individual nodes has a negligible spillover effect on the expectation of $\mathbf{a}_i$.*

Algorithm 1 The Proposed Method **DeCaf**

Input: node features $\mathbf{X} \in \mathbb{R}^{n \times d}$, unnormalized adjacency matrix $A \in \{0,1\}^{n \times n}$
Output: predicted class labels $\mathbf{Y} \in \mathbb{I}^{n \times k}$
while not converged **do**
    Evaluate $J_{p,p'}(\rho, \rho')$ based on Equation 20
    Update $\rho \leftarrow \rho - \hat{\Delta}_{\rho} J_{p,p'}(\rho, \rho')$
    Update $\rho' \leftarrow \rho' - \hat{\Delta}_{\rho'} J_{p,p'}(\rho, \rho')$
end while
Evaluate $\mathbf{h}_i^{\mathbf{a}}$ and $\mathbf{m}_i^{\mathbf{a}}$
while not converged **do**
    Sample mini-batch $\{\mathbf{x}_i, \mathbf{h}_i^{\mathbf{a}}\}_{i=1}^{b}$
    Evaluate $J_g(\psi^A)$
    Update $\theta^A \leftarrow \theta^A - \hat{\Delta}_{\theta^A} J_m(\theta^A)$
end while
while not converged **do**
    Sample mini-batch $\{\mathbf{x}_i, \mathbf{h}_i^{\mathbf{a}}\}_{i=1}^{b}$
    Evaluate $J_m(\theta^A)$, $J_e(\eta^X)$
    for step = 1 **to** *step_size* **do**
        Update $\theta^A \leftarrow \theta^A - \hat{\Delta}_{\theta^A} J_m(\theta^A)$
    end for
    Update $\eta^A \leftarrow \eta^A - \hat{\Delta}_{\eta^A} J_e(\eta^X)$
end while
while not converged **do**
    Sample mini-batch $\{\mathbf{x}_i, \mathbf{h}_i^{\mathbf{a}}, \mathbf{m}_i^{\mathbf{a}}\}_{i=1}^{b}$
    Evaluate $J_h(\phi^X)$, $J_e(\eta^X)$
    for step = 1 **to** *step_size* **do**
        Update $\phi^X \leftarrow \phi^X - \hat{\Delta}_{\phi^X} J_m(\phi^X)$
    end for
    Update $\eta^X \leftarrow \eta^X - \hat{\Delta}_{\eta^X} J_e(\eta^X)$
end while
Evaluate $\mathbf{E}[\mathbf{y}|C = \mathbf{a}, do(T = \mathbf{x}')]$, $\mathbf{E}[\mathbf{y}|C = \mathbf{x}, do(T = \mathbf{a}')]$ based on Equations in Section 2.4.2
Evaluate $\Psi_{\mathbf{a}}(\mathbf{a})$, $\Psi_{\mathbf{x}}(\mathbf{x})$ based on Equation 7 and 8
Evaluate $\mathbf{y}$ based on Equation 14

Table 5: Important notations and descriptions.

Notation	Description
n, d, k, o	# nodes, feature size, # classes, hidden size
$\mathbf{x}$, $\mathbf{x}_i$, $\mathbf{X} \in \mathbb{R}^{n \times d}$	node feature vector, node feature of i-th instance, node feature matrix
$\mathbf{y}$, $\mathbf{y}_i$, $\mathbf{Y} \in \mathbb{I}^{n \times k}$	node label vector, label of i-th instance, node label matrix
$\mathbf{a}$, $\mathbf{a}_i$, $\mathbf{A} \in \mathbb{R}^{n \times o}$	neighborhood representation, neighborhood representation, of i-th instance, neighborhood representation matrix
$A \in \mathbb{I}^{n \times n}$	unnormalized adjacency matrix
$p(\cdot), p'(\cdot)$	GNN embedding layer, MLP layer
ρ, ρ'	parameters of $p(\cdot), p'(\cdot)$
$\mathbf{h}_i^{\mathbf{a}}$, $\mathbf{m}_i^{\mathbf{a}}$	hidden neighborhood representation of node i, and its project to output space
$g^X(\cdot), g^A(\cdot)$	confounder representation function for SCM-F and SCM-A
ψ^X, ψ^A	parameters for $g^X(\cdot), g^A(\cdot)$
$h^X(\cdot), h^A(\cdot)$	treatment representation function for SCM-F and SCM-A
ϕ^X, ϕ^A	parameters for $h^X(\cdot), h^A(\cdot)$
$m^X(\cdot), m^A(\cdot)$	confounder predicting function for SCM-F and SCM-A
θ^X, θ^A	parameters for $m^X(\cdot), m^A(\cdot)$
$e^X(\cdot), e^A(\cdot)$	propensity feature function for SCM-F and SCM-A
η^X, η^A	parameters for $e^X(\cdot), e^A(\cdot)$
$\mathbf{x}'$, $\mathbf{a}'$	counterfactual node feature and neighborhood representation

E ANALYSIS ON DISTRIBUTION SHIFT

Proposition 1. *Under covariate shift, the correlation between* $\mathbf{x}$ *and* $\mathbf{a}$ *shifts (e.g.* $P^{train}(\mathbf{a}|\mathbf{x}) \neq P^{test}(\mathbf{a}|\mathbf{x})$ *or* $P^{train}(\mathbf{x}|\mathbf{a}) \neq P^{test}(\mathbf{x}|\mathbf{a})$*). Consequently, the conditional distribution of* $\mathbf{y}$ *given both* $\mathbf{x}$ *and* $\mathbf{a}$ *shifts (e.g.* $P^{train}(\mathbf{y}|\mathbf{x}, \mathbf{a}) \neq P^{test}(\mathbf{y}|\mathbf{x}, \mathbf{a})$*). However, the conditional distribution of* $\mathbf{y}$ *given* $\mathbf{x}$ *alone remain invariant (e.g.* $P^{train}(\mathbf{y}|\mathbf{x}) = P^{test}(\mathbf{y}|\mathbf{x})$*).*

Proof. Under covariate shift, we have:

$$P^{\text{train}}(\mathbf{z}) \neq P^{\text{test}}(\mathbf{z})$$

while the transformation matrices and bias vectors for generating $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{a}$ remain the same between training and testing datasets:

$$\mathcal{M}_x^{\text{train}} = \mathcal{M}_x^{\text{test}}, \quad \mathbf{b}_x^{\text{train}} = \mathbf{b}_x^{\text{test}}$$
$$\mathcal{M}_y^{\text{train}} = \mathcal{M}_y^{\text{test}}, \quad \mathbf{b}_y^{\text{train}} = \mathbf{b}_y^{\text{test}}$$
$$\mathcal{M}_s^{\text{train}} = \mathcal{M}_s^{\text{test}}, \quad \mathcal{M}_o^{\text{train}} = \mathcal{M}_o^{\text{test}}$$

The generation processes are:

$$\mathbf{x}_i = \mathcal{M}_x \mathbf{z}_i + \mathbf{b}_x$$
$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$$
$$p(A_{ij} = 1) = c\left(\|\mathcal{M}_s \mathbf{z}_i - \mathcal{M}_o \mathbf{z}_j\|_2^2 + 1\right)^{-1}$$

where $\mathbf{a}_i$ is defined as:

$$\mathbf{a}_i = \text{agg}(\{\mathcal{M}_a \mathbf{z}_j\}) \quad \text{where} \quad j \in \mathcal{N}_i^1 \cup \mathcal{N}_i^2 \cup \ldots \cup \mathcal{N}_i^l$$

Given that $P(\mathbf{z})$ changes, the marginal distribution of $\mathbf{x}$ changes accordingly:

$$P^{\text{train}}(\mathbf{x}) \neq P^{\text{test}}(\mathbf{x})$$

Since $\mathbf{a}_i$ is an aggregation of transformed neighbors' latent vectors $\mathbf{z}_j$, and $P(\mathbf{z})$ changes, the distribution of $\mathbf{a}_i$ also changes:

$$P^{\text{train}}(\mathbf{a}) \neq P^{\text{test}}(\mathbf{a})$$

To show that the joint distribution $P(\mathbf{x}, \mathbf{a})$ changes, consider that $\mathbf{a}$ is derived from $\mathbf{z}$ through a different transformation $\mathcal{M}_a$ and aggregation function. Since $\mathbf{x}$ and $\mathbf{a}$ are both functions of $\mathbf{z}$, any change in the distribution of $\mathbf{z}$ will induce changes in both $P(\mathbf{x})$ and $P(\mathbf{a})$. Given that $P(\mathbf{x})$ and $P(\mathbf{a})$ change independently due to the change in $P(\mathbf{z})$, the joint distribution $P(\mathbf{x}, \mathbf{a})$ must also change because the dependencies between $\mathbf{x}$ and $\mathbf{a}$ are functions of $\mathbf{z}$:

$$P^{\text{train}}(\mathbf{x}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{x}, \mathbf{a})$$

This implies that:

$$P^{\text{train}}(\mathbf{a}|\mathbf{x}) = \frac{P^{\text{train}}(\mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{x})} \neq \frac{P^{\text{test}}(\mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{x})} = P^{\text{test}}(\mathbf{a}|\mathbf{x})$$

$$P^{\text{train}}(\mathbf{x}|\mathbf{a}) = \frac{P^{\text{train}}(\mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{a})} = P^{\text{test}}(\mathbf{x}|\mathbf{a})$$

The generation process for $\mathbf{y}$ is:

$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$$

Since $\mathcal{M}_y$ and $\mathbf{b}_y$ are the same across training and testing, and $\mathbf{z}_i$ influences $\mathbf{x}_i$ and $\mathbf{a}_i$ through $\mathcal{M}_x$, $\mathcal{M}_s$, $\mathcal{M}_o$, and $\mathcal{M}_a$, the change in $P(\mathbf{z})$ affects $P(\mathbf{x})$ and $P(\mathbf{a})$. Consequently, the joint distribution $P(\mathbf{y}, \mathbf{x}, \mathbf{a})$ changes:

$$P^{\text{train}}(\mathbf{y}, \mathbf{x}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{y}, \mathbf{x}, \mathbf{a})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{x}, \mathbf{a}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{x}, \mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{y}, \mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{x}, \mathbf{a})} = P^{\text{test}}(\mathbf{y}|\mathbf{x}, \mathbf{a})$$

Since $\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$ and $\mathcal{M}_y$ and $\mathbf{b}_y$ are invariant, the conditional distribution $P(\mathbf{y}|\mathbf{z})$ remains invariant:

$$P^{\text{train}}(\mathbf{y}|\mathbf{z}) = P^{\text{test}}(\mathbf{y}|\mathbf{z})$$

Given that $\mathbf{x}_i = \mathcal{M}_x \mathbf{z}_i + \mathbf{b}_x$ and $\mathcal{M}_x$ and $\mathbf{b}_x$ are invariant, the conditional distribution $P(\mathbf{y}|\mathbf{x})$ also remains invariant:

$$P^{\text{train}}(\mathbf{y}|\mathbf{x}) = P^{\text{test}}(\mathbf{y}|\mathbf{x})$$

□

Proposition 2. *Under concept shift-F, the dependencies between* $\mathbf{x}$ *and* $\mathbf{a}$ *shifts (e.g.* $P^{train}(\mathbf{a}|\mathbf{x}) \neq P^{test}(\mathbf{a}|\mathbf{x})$ *or* $P^{train}(\mathbf{x}|\mathbf{a}) \neq P^{test}(\mathbf{x}|\mathbf{a})$*), and the conditional distribution of* Y *given* F *shifts (e.g.* $P^{train}(\mathbf{y}|\mathbf{x}) \neq P^{test}(\mathbf{y}|\mathbf{x})$*). Consequently, the conditional distribution of* $\mathbf{y}$ *given both* $\mathbf{x}$ *and* $\mathbf{a}$ *shifts (e.g.* $P^{train}(\mathbf{y}|\mathbf{x}, \mathbf{a}) \neq P^{test}(\mathbf{y}|\mathbf{x}, \mathbf{a})$*). However, the conditional distribution of* Y *given* A *alone remain invariant (e.g.* $P^{train}(\mathbf{y}|\mathbf{a}) = P^{test}(\mathbf{y}|\mathbf{a})$*).*

Proof. Under concept shift-F, we have:

$$P^{\text{train}}(\mathbf{z}) = P^{\text{test}}(\mathbf{z})$$

while the transformation matrices and bias vectors for generating $\mathbf{x}$ change, but those for $\mathbf{a}$ and $\mathbf{y}$ remain the same between training and testing datasets:

$$\mathcal{M}_x^{\text{train}} \neq \mathcal{M}_x^{\text{test}}, \quad \mathbf{b}_x^{\text{train}} \neq \mathbf{b}_x^{\text{test}}$$

$$\mathcal{M}_y^{\text{train}} = \mathcal{M}_y^{\text{test}}, \quad \mathbf{b}_y^{\text{train}} = \mathbf{b}_y^{\text{test}}$$

$$\mathcal{M}_s^{\text{train}} = \mathcal{M}_s^{\text{test}}, \quad \mathcal{M}_o^{\text{train}} = \mathcal{M}_o^{\text{test}}$$

The generation processes are:

$$\mathbf{x}_i^{\text{train}} = \mathcal{M}_x^{\text{train}} \mathbf{z}_i + \mathbf{b}_x^{\text{train}}$$

$$\mathbf{x}_i^{\text{test}} = \mathcal{M}_x^{\text{test}} \mathbf{z}_i + \mathbf{b}_x^{\text{test}}$$

$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$$
$$p(A_{ij} = 1) = c\left(\|\mathcal{M}_s \mathbf{z}_i - \mathcal{M}_o \mathbf{z}_j\|_2^2 + 1\right)^{-1}$$

where $\mathbf{a}_i$ is defined as:

$$\mathbf{a}_i = \text{agg}(\{\mathcal{M}_a \mathbf{z}_j\}) \quad \text{where} \quad j \in \mathcal{N}_i^1 \cup \mathcal{N}_i^2 \cup \ldots \cup \mathcal{N}_i^l$$

Given that $P(\mathbf{z})$ remains the same, the marginal distribution of $\mathbf{a}$ does not change:

$$P^{\text{train}}(\mathbf{a}) = P^{\text{test}}(\mathbf{a})$$

However, the change in the transformation matrix $\mathcal{M}_x$ and bias vector $\mathbf{b}_x$ implies that the marginal distribution of $\mathbf{x}$ changes:

$$P^{\text{train}}(\mathbf{x}) \neq P^{\text{test}}(\mathbf{x})$$

Since $\mathbf{a}$ is an aggregation of transformed neighbors' latent vectors $\mathbf{z}_j$, and the generation process for $\mathbf{a}$ remains unchanged, the joint distribution $P(\mathbf{x}, \mathbf{a})$ changes due to the change in $P(\mathbf{x})$:

$$P^{\text{train}}(\mathbf{x}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{x}, \mathbf{a})$$

This implies that:

$$P^{\text{train}}(\mathbf{a}|\mathbf{x}) = \frac{P^{\text{train}}(\mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{x})} \neq \frac{P^{\text{test}}(\mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{x})} = P^{\text{test}}(\mathbf{a}|\mathbf{x})$$
$$P^{\text{train}}(\mathbf{x}|\mathbf{a}) = \frac{P^{\text{train}}(\mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{a})} = P^{\text{test}}(\mathbf{x}|\mathbf{a})$$

The generation process for $\mathbf{y}$ is:

$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$$

Since $\mathcal{M}_y$ and $\mathbf{b}_y$ are the same across training and testing, and $\mathbf{z}_i$ influences $\mathbf{x}_i$ through $\mathcal{M}_x$, the change in $\mathcal{M}_x$ affects the joint distribution $P(\mathbf{y}, \mathbf{x})$:

$$P^{\text{train}}(\mathbf{y}, \mathbf{x}) \neq P^{\text{test}}(\mathbf{y}, \mathbf{x})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{x}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{x})}{P^{\text{train}}(\mathbf{x})} \neq \frac{P^{\text{test}}(\mathbf{y}, \mathbf{x})}{P^{\text{test}}(\mathbf{x})} = P^{\text{test}}(\mathbf{y}|\mathbf{x})$$

The change in the joint distribution $P(\mathbf{x}, \mathbf{a})$ implies a change in the joint distribution $P(\mathbf{y}, \mathbf{x}, \mathbf{a})$ since $\mathbf{y}$ depends on both $\mathbf{x}$ and $\mathbf{a}$:

$$P^{\text{train}}(\mathbf{y}, \mathbf{x}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{y}, \mathbf{x}, \mathbf{a})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{x}, \mathbf{a}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{x}, \mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{y}, \mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{x}, \mathbf{a})} = P^{\text{test}}(\mathbf{y}|\mathbf{x}, \mathbf{a})$$

Since $\mathbf{y}$ and $\mathbf{a}$ are both derived from $\mathbf{z}$ through invariant transformation matrices and bias vectors, and $P(\mathbf{z})$ remains the same, the joint distribution $P(\mathbf{y}, \mathbf{a})$ remains unchanged:

$$P^{\text{train}}(\mathbf{y}, \mathbf{a}) = P^{\text{test}}(\mathbf{y}, \mathbf{a})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{a}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{a})}{P^{\text{train}}(\mathbf{a})} = \frac{P^{\text{test}}(\mathbf{y}, \mathbf{a})}{P^{\text{test}}(\mathbf{a})} = P^{\text{test}}(\mathbf{y}|\mathbf{a})$$

□

Proposition 3. *Under concept shift-A, the dependencies between* $\mathbf{x}$ *and* $\mathbf{a}$ *shifts (e.g.* $P^{train}(\mathbf{a}|\mathbf{x}) \neq P^{test}(\mathbf{a}|\mathbf{x})$ *or* $P^{train}(\mathbf{x}|\mathbf{a}) \neq P^{test}(\mathbf{x}|\mathbf{a})$*), and the conditional distribution of* $\mathbf{y}$ *given A shifts (e.g.* $P^{train}(\mathbf{y}|\mathbf{a}) \neq P^{test}(\mathbf{y}|\mathbf{a})$*). Consequently, the conditional distribution of* Y *given both* $\mathbf{x}$ *and* $\mathbf{a}$ *shifts (e.g.* $P^{train}(\mathbf{y}|\mathbf{x},\mathbf{a}) \neq P^{test}(\mathbf{y}|\mathbf{x},\mathbf{a})$*). However, the conditional distribution of* $\mathbf{y}$ *given* $\mathbf{x}$ *alone remain invariant (e.g.* $P^{train}(\mathbf{y}|\mathbf{x}) = P^{test}(\mathbf{y}|\mathbf{x})$*).*

Proof. Under concept shift-A, we have:

$$P^{\text{train}}(\mathbf{z}) = P^{\text{test}}(\mathbf{z})$$

while the transformation matrices and bias vectors for generating $\mathbf{a}$ change, but those for $\mathbf{x}$ and $\mathbf{y}$ remain the same between training and testing datasets:

$$\mathcal{M}_x^{\text{train}} = \mathcal{M}_x^{\text{test}}, \quad \mathbf{b}_x^{\text{train}} = \mathbf{b}_x^{\text{test}}$$
$$\mathcal{M}_y^{\text{train}} = \mathcal{M}_y^{\text{test}}, \quad \mathbf{b}_y^{\text{train}} = \mathbf{b}_y^{\text{test}}$$
$$\mathcal{M}_s^{\text{train}} \neq \mathcal{M}_s^{\text{test}}, \quad \mathcal{M}_o^{\text{train}} \neq \mathcal{M}_o^{\text{test}}$$

The generation processes are:

$$\mathbf{x}_i = \mathcal{M}_x \mathbf{z}_i + \mathbf{b}_x$$
$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$$
$$p^{\text{train}}(A_{ij} = 1) = c\left(\|\mathcal{M}_s^{\text{train}}\mathbf{z}_i - \mathcal{M}_o^{\text{train}}\mathbf{z}_j\|_2^2 + 1\right)^{-1}$$
$$p^{\text{test}}(A_{ij} = 1) = c\left(\|\mathcal{M}_s^{\text{test}}\mathbf{z}_i - \mathcal{M}_o^{\text{test}}\mathbf{z}_j\|_2^2 + 1\right)^{-1}$$

where $\mathbf{a}_i$ is defined as:

$$\mathbf{a}_i = \text{agg}(\{\mathcal{M}_a \mathbf{z}_j\}) \quad \text{where} \quad j \in \mathcal{N}_i^1 \cup \mathcal{N}_i^2 \cup \ldots \cup \mathcal{N}_i^l$$

Given that $P(\mathbf{z})$ remains the same, the marginal distribution of $\mathbf{x}$ does not change:

$$P^{\text{train}}(\mathbf{x}) = P^{\text{test}}(\mathbf{x})$$

However, the change in the transformation matrices $\mathcal{M}_s$ and $\mathcal{M}_o$ implies that the marginal distribution of $\mathbf{a}$ changes:

$$P^{\text{train}}(\mathbf{a}) \neq P^{\text{test}}(\mathbf{a})$$

Since $\mathbf{a}$ is an aggregation of transformed neighbors' latent vectors $\mathbf{z}_j$, and the generation process for $\mathbf{a}$ changes, the joint distribution $P(\mathbf{x}, \mathbf{a})$ changes due to the change in $P(\mathbf{a})$:

$$P^{\text{train}}(\mathbf{x}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{x}, \mathbf{a})$$

This implies that:

$$P^{\text{train}}(\mathbf{a}|\mathbf{x}) = \frac{P^{\text{train}}(\mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{x})} \neq \frac{P^{\text{test}}(\mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{x})} = P^{\text{test}}(\mathbf{a}|\mathbf{x})$$
$$P^{\text{train}}(\mathbf{x}|\mathbf{a}) = \frac{P^{\text{train}}(\mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{a})} = P^{\text{test}}(\mathbf{x}|\mathbf{a})$$

The generation process for $\mathbf{y}$ is:

$$\mathbf{y}_i = \mathcal{M}_y \mathbf{z}_i + \mathbf{b}_y$$

Since $\mathcal{M}_y$ and $\mathbf{b}_y$ are the same across training and testing, and $\mathbf{z}_i$ influences $\mathbf{a}_i$ through $\mathcal{M}_s$ and $\mathcal{M}_o$, the change in $\mathcal{M}_s$ and $\mathcal{M}_o$ affects the joint distribution $P(\mathbf{y}, \mathbf{a})$:

$$P^{\text{train}}(\mathbf{y}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{y}, \mathbf{a})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{a}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{a})}{P^{\text{train}}(\mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{y}, \mathbf{a})}{P^{\text{test}}(\mathbf{a})} = P^{\text{test}}(\mathbf{y}|\mathbf{a})$$

Since $\mathbf{y}$ and $\mathbf{x}$ are both derived from $\mathbf{z}$ through invariant transformation matrices and bias vectors, and $P(\mathbf{z})$ remains the same, the joint distribution $P(\mathbf{y}, \mathbf{x})$ remains unchanged:

$$P^{\text{train}}(\mathbf{y}, \mathbf{x}) = P^{\text{test}}(\mathbf{y}, \mathbf{x})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{x}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{x})}{P^{\text{train}}(\mathbf{x})} = \frac{P^{\text{test}}(\mathbf{y}, \mathbf{x})}{P^{\text{test}}(\mathbf{x})} = P^{\text{test}}(\mathbf{y}|\mathbf{x})$$

The change in the joint distribution $P(\mathbf{x}, \mathbf{a})$ implies a change in the joint distribution $P(\mathbf{y}, \mathbf{x}, \mathbf{a})$ since $\mathbf{y}$ depends on both $\mathbf{x}$ and $\mathbf{a}$:

$$P^{\text{train}}(\mathbf{y}, \mathbf{x}, \mathbf{a}) \neq P^{\text{test}}(\mathbf{y}, \mathbf{x}, \mathbf{a})$$

Thus, the conditional distribution:

$$P^{\text{train}}(\mathbf{y}|\mathbf{x}, \mathbf{a}) = \frac{P^{\text{train}}(\mathbf{y}, \mathbf{x}, \mathbf{a})}{P^{\text{train}}(\mathbf{x}, \mathbf{a})} \neq \frac{P^{\text{test}}(\mathbf{y}, \mathbf{x}, \mathbf{a})}{P^{\text{test}}(\mathbf{x}, \mathbf{a})} = P^{\text{test}}(\mathbf{y}|\mathbf{x}, \mathbf{a})$$

□

F IMPORTANT ASSUMPTIONS FOR CATE ESTIMATION

Assumption 5. *(SUTVA). The potential outcomes of any unit do not vary with the treatment assigned to other units, and, for each unit, there are no different forms or versions of each treatment level, which leads to different potential outcomes.*

We discuss the reasonableness of this assumption in section 2.1.

Assumption 6. *(Consistency). The potential outcome of treatment t equals the observed outcome if the actual treatment received is t.*

Assumption 7. *(Ignorability). Given pretreatment covariate $\mathbf{X}$, the outcome variable Y_0 and Y_1 is independent of treatment assignment, i.e. $(Y_0, Y_1 \perp\!\!\!\perp T|X)$.*

This assumption is also called "*no unmeasured confounder*". This assumption is automatically satisfied with the *"close-world assumption"* made in learning a machine learning model, which implicitly assumes that the input data encompasses the necessary information for making accurate predictions, as we explain in section 2.1. In our case, it implies that no other confounders besides $\mathbf{a}$ and $\mathbf{x}$ that affect the output should exist.

Assumption 8. *(Positivity). For any set of covariates $\mathbf{x}$, the probability to receive any treatment t is positive, i.e., $0 < P(T = t|X = x) < 1, \forall t, x$.*

G DERIVATION OF THE PREDICTED FUNCTION OF SGC

A typical SGC makes predictions with the classifier:

$$\hat{y} = \sigma(S^k \mathbf{X} \Theta), \tag{15}$$

where S is the "normalized" adjacency matrix $S = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}}$, $\tilde{A} = A + I$, $\tilde{D}$ is the degree matrix of $\tilde{A}$, k is the number of layers, Θ is the parameterized weights of each layer into a single matrix: $\Theta = \Theta_0 \Theta_1 ... \Theta_k$. Note that the above equation averages the representation of all nodes at each hop, so the effect of the central node is diminished when its neighbor size is large. Alternatively, we can assign a fixed weight $0 < \gamma < 1$ to the central node, and the rest $1 - \gamma$ is shared by the neighboring nodes during the aggregation process, so the S^k in equation is replaced by S', where:

$$S' = \tilde{D}_1^{-\frac{1}{2}} A \tilde{A}^k \tilde{D}_1^{\frac{1}{2}} + \tilde{D}_2^{-\frac{1}{2}} I \tilde{D}_2^{\frac{1}{2}}, \tag{16}$$

where $\tilde{D}_1$ and $\tilde{D}_2$ are diagonal matrix such that $\tilde{D}_1(i,i) = (1-\gamma)D^k(i,i)$ and $\tilde{D}_2(i,i) = \gamma D^k(i,i)$. The new prediction function is then expressed as:

$$\hat{\mathbf{y}}_i = \sigma(\mathbf{h}_i^y) = \sigma([\mathbf{S'}^k\mathbf{X}\Theta]_i) \tag{17}$$

$$= \sigma\Big(\underbrace{[\tilde{D_1}^{-\frac{1}{2}}A\tilde{A}^k\tilde{D}_1^{\frac{1}{2}}\mathbf{X}\Theta]_i}_{\Psi'_{\mathbf{a}}(\mathbf{a}_i)} + \underbrace{[(\tilde{D_2}^{-\frac{1}{2}}I\tilde{D}_2^{\frac{1}{2}})^k\mathbf{X}\Theta]_i}_{\Psi'_{\mathbf{x}}(\mathbf{x_i})}\Big). \tag{18}$$

Inside $\sigma(\cdot)$, the first term $\tilde{D_1}^{-\frac{1}{2}}A\tilde{A}^k\tilde{D}_1^{\frac{1}{2}}X\Theta$ models the contribution of the neighborhood nodes' representation (excluding central node) on $\mathbf{h}_i^y$, we call it $\Psi'_{\mathbf{a}}(\mathbf{a}_i)$; similarly, the second term $(\tilde{D_2}^{-\frac{1}{2}}I\tilde{D}_2^{\frac{1}{2}})^kX\Theta$ models the contribution of the features of the central node and we call it $\Psi'_{\mathbf{x}}(\mathbf{x}_i)$. Note that the magnitudes of the diagnal matrixes of $\Psi'_{\mathbf{a}}(\mathbf{a}_i)$ and $\Psi'_{\mathbf{x}}(\mathbf{x}_i)$ are scaled by the parameter γ. We can further rewrite Equation 18 in the unscaled form:

$$\sigma\Big(\gamma\underbrace{[(\tilde{D}^k)^{-\frac{1}{2}}A\tilde{A}^k(\tilde{D}^k)^{\frac{1}{2}}\mathbf{X}\Theta]_i}_{\Psi_{\mathbf{a}}(\mathbf{a}_i)} + (1-\gamma)\underbrace{[((\tilde{D}^k)^{-\frac{1}{2}}I(\tilde{D}^k)^{\frac{1}{2}})^k\mathbf{X}\Theta]_i}_{\Psi_{\mathbf{x}}(\mathbf{x_i})}\Big), \tag{19}$$

where $[(\tilde{D}^k)^{-\frac{1}{2}}A\tilde{A}^k(\tilde{D}^k)^{\frac{1}{2}}\mathbf{X}\Theta]_i$ can be viewed as $\Psi_{\mathbf{a}}(\mathbf{a}_i)$, and $[((\tilde{D}^k)^{-\frac{1}{2}}I(\tilde{D}^k)^{\frac{1}{2}})^k\mathbf{X}\Theta]_i$ can be viewed as $\Psi_{\mathbf{x}}(\mathbf{x_i})$, which aligns with asssumtion 2.

H DUAL CAUSAL DECOMPOSITION

We aim to learn two sets of $g(\cdot), h(\cdot)$ for the two casual models SCM-F and SCM-A. We denote them as $g^X(\cdot), h^X(\cdot)$ and $g^A(\cdot), h^A(\cdot)$, separately. Each casual model is also associated with a set of $e(\cdot), m(\cdot)$, denoted as $e^X(\cdot), m^X(\cdot)$ and $e^A(\cdot), m^A(\cdot)$. Unlike SIN, which learns all model parameters within the two-stage training procedure, we allow the model to learn $g^X(\cdot)$, $h^A(\cdot)$, and $m^X(\cdot)$ with shared parameters beforehand. Since $g^X(\mathbf{a})$, $h^A(\mathbf{a})$, and $m^X(\mathbf{a})$ are all functions of the neighborhood representation $\mathbf{a}$, whose value is determined with a L-layer GNN model $p^{GNN}(\cdot)$ that generates a neighborhood representation by aggregating the embedding of nodes in L-hop neighborhood without including the central node.

We then map the GNN embedding to the space of $\mathbf{h}^y$ with an MLP layer $p'(\cdot)$ that follows $p(\cdot)$.

The parameters of $p(\cdot)$ and $p'(\cdot)$, ρ, ρ', are learned with the goal of minimizing: $J_{p,p'}(\rho,\rho') = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{\text{mse}}\left(\mathbf{h}_i^y, \left[\hat{p'}_{\rho'}(\hat{p}_\rho(\mathbf{X}, A))\right]_i\right)$. As the ground truth $\mathbf{h}^y$ is not available, while $\mathbf{y} = \sigma(\mathbf{h}^y)$ is available, we thus apply $\sigma(\cdot)$ on both sides and minimizing the following cross entropy function instead:

$$J_{p,p'}(\rho,\rho') = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{ce}\left(\mathbf{y}_i, \sigma\Big(\left[\hat{p'}_{\rho'}(\hat{p}_\rho(\mathbf{X}, A))\right]_i\Big)\right). \tag{20}$$

We apply this alternation for the rest of the training process. With the optimized $p(\cdot)$, we first assign the learned neighborhood representation to $\mathbf{a}$, such that $\mathbf{a}_i \triangleq [\hat{p}_\rho(\mathbf{X}, A)]_i$. Without losing generalizability, we assign $g^X(\mathbf{a})$, $h^A(\mathbf{a})$ as the same as $\mathbf{a}$. We then estimate $m^X(\mathbf{a})$, with the optimized $p'(\cdot)$. $g^X(\mathbf{a})$, $h^A(\mathbf{a})$, $m^X(\mathbf{a})$ and remain fixed values in the rest of the learning process: $g^X(\mathbf{a}_i) \triangleq h^A(\mathbf{a}_i) \triangleq \mathbf{h}_i^{\mathbf{a}} \triangleq \mathbf{a}_i, m^X(\mathbf{a}_i) \triangleq \mathbf{m}_i^{\mathbf{a}} \triangleq \hat{p'}_{\rho'}(\mathbf{a}_i)$

We learn the remaining parameters for each casual model. For SCM-A, we follow the two-stage procedure:

Stage 1: Learn parameter θ^A of $\hat{m}_\theta(\mathbf{x})$ to minimize the cross-entropy loss as following: $J_m(\theta^A) = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{ce}\left(\mathbf{y}_i, \sigma\big(\hat{m}_{\theta^A}(\mathbf{x}_i)\big)\right)^2$

Stage 2: Learn parameter ψ^A for $g^A(\cdot)$ and η^A for $e^A(\cdot)$ with the objectives:

$$J_g(\psi^A) = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{ce}\left(\mathbf{y}_i, \sigma\left(\hat{m}_{\theta^A}(\mathbf{x}_i) + \hat{g}_{\psi^A}(\mathbf{x}_i)^\top\big(\mathbf{h}_i^{\mathbf{a}} - \hat{e}_{\eta^A}^A(\mathbf{x}_i)\big)\right)\right)^2, J_e(\eta^A) = \frac{1}{n}\sum_{i=1}^n \left\|\mathbf{h}_i^{\mathbf{a}} - \hat{e}_{\eta^A}^A(\mathbf{x}_i)\right\|_2^2 \tag{21}$$

For SCM-F, stage 1 is no longer necessary as $\hat{m}_\theta(\mathbf{a}_i)$ is fixed as $\mathbf{m}_i^{\mathbf{a}}$. We only need to learn parameter ϕ^X for

$h^X(\cdot)$ and η_F for $e^X(\cdot)$ with the objectives:

$$J_h(\phi^X) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{ce}\left(\mathbf{y}_i, \sigma\left(\mathbf{m}_i^{\mathbf{a}} + \mathbf{h}_i^{\mathbf{a}\top}\left(\hat{h}_{\phi^X}^X(\mathbf{x}_i) - \hat{e}_{\eta^X}^A(\mathbf{a}_i)\right)\right)\right)^2, J_e(\eta^X) = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{h}^A - \hat{e}_{\eta^X}^A(\mathbf{h}_i^{\mathbf{a}})\right\|_2^2 \tag{22}$$

We follow the alternating optimization process in [Kaddour et al., 2021] which updates ψ, ϕ more frequently than η to achieve a more stabilized training process.

I COMPLETE RESULTS

We report the complete results on `facebook-100` when different sets of training graphs are used in Table 6. With different training sets, the proposed **DeCaf** achieves the best classification results over three different test sets compared with state-of-the-art OOD generalization methods.

Table 6: Test F1 scores on `Facebook-100` dataset. The results from the GNN with the highest validation F1 score are reported. The best results are bold-faced.

Training	Johns Hopkins + Caltech + Amherst		
Test	Penn	Brown	Texas
ERM	49.23±1.72	49.68±0.93	48.57±0.21
IRM	35.26±2.40	46.92±5.66	36.86±1.64
REX	44.77±6.48	42.65±7.34	44.05±8.88
EERM	22.62±22.91	49.44±1.92	49.12±1.71
CIT	44.66±6.65	45.26±6.21	42.10±8.97
FLOOD	42.37±5.06	41.48±5.28	40.82±5.94
StableGL	44.54±6.58	45.64±6.25	43.75±5.65
DeCaf	**55.31±0.40**	**53.31±0.11**	**53.56±0.19**
Training	Bingham + Duke + Princeton		
Test	Penn	Brown	Texas
ERM	51.42±4.25	51.45±1.48	47.37±4.78
IRM	42.12±1.99	51.34±0.90	41.57±4.31
REX	43.77±5.72	47.26±5.75	44.36±7.82
EERM	18.91±18.99	45.95±3.74	47.83±1.17
CIT	45.82±6.46	48.62±1.88	39.79±4.74
FLOOD	46.99±7.48	47.28±5.61	44.48±5.22
StableGL	46.94±8.54	47.77±4.79	47.51±5.94
DeCaf	**54.59±0.35**	**53.48±0.15**	**53.12±0.19**
Training	WashU + Brandeis + Carnegie		
Test	Penn	Brown	Texas
ERM	47.34±5.48	48.08±2.51	48.36±4.30
IRM	50.16±1.30	49.62±3.32	46.41±5.28
REX	39.67±8.59	44.65±6.67	40.28±7.59
EERM	24.40±24.62	47.58±2.91	51.27±1.04
CIT	37.99±6.54	39.66±6.58	39.88±6.27
FLOOD	41.21±7.68	46.24±8.52	40.16±6.01
StableGL	38.31±8.42	43.22±4.50	37.86±4.75
DeCaf	**54.44±1.18**	**53.02±0.36**	**53.05±0.86**

J COMPLEXITY ANALYSIS

Consider a graph with n nodes and e edges, and an average degree $\bar{d}$. A Graph Neural Network (GNN) with L layers computes embeddings with a time and space complexity of $O\left(nL\bar{d}^2\right)$. When obtaining the GNN embedding, **DeCaf** performs one encoder computation per update step. During training, five distinct encoders are learned for causal models, each with a time complexity of $O(n)$ per update step. Therefore, the overall time complexity is $O\left(nL\bar{d}^2\right) + O\left(5n\right)$.

We compare the complexity and requirements of **DeCaf** with other methods in Table 8. For EERM and FLOOD, K is the number of augmented training environments. For CIT, K is the number of clusters and p is the probability of transfer. As it shows, **DeCaf** has competitive complexity, while having the least restrictive requirements, making it applicable to a wide range of scenarios.

Table 7: Test Macro-F1 scores on synthetic datasets. The best results are bold-faced.

		h-feat	qrt-feat	full-feat
SGC	ERM	47.41±0.25	48.62±1.88	48.57±0.21
	IRM	33.78±1.41	33.36±0.70	33.04±1.40
	EERM	38.14±8.25	37.86±5.15	39.74±7.09
	CIT	21.11±21.33	36.41±0.45	49.12±1.71
	REX	32.87±1.12	33.85±0.50	34.47±0.82
	FLOOD	39.53±8.83	37.26±4.82	39.13±6.39
	StableGL	41.79±8.86	38.30±5.60	41.22±6.30
	DeCaf	**55.31±0.40**	**53.16±0.17**	**53.56±0.19**
GCN	ERM	43.12±1.84	49.68±0.93	43.79±1.36
	IRM	34.03±0.94	32.90±0.60	32.47±1.16
	EERM	43.24±6.52	42.65±7.34	38.60±5.26
	CIT	22.79±23.15	49.44±1.92	40.81±5.74
	REX	33.10±1.60	35.62±2.60	34.49±0.89
	FLOOD	38.58±6.60	40.07±6.76	38.58±6.57
	StableGL	42.12±6.44	44.77±4.78	43.75±5.65
	DeCaf	**53.57±0.98**	**53.31±0.11**	**52.16±0.45**
GAT	ERM	49.23±1.72	47.34±1.25	46.13±1.54
	IRM	35.26±2.40	46.92±5.66	36.86±1.64
	EERM	44.77±6.48	41.94±3.92	44.05±8.88
	CIT	22.62±22.91	39.04±4.31	34.66±2.32
	REX	44.66±6.65	45.26±6.21	42.10±8.97
	FLOOD	42.37±5.06	41.48±5.28	40.82±5.94
	StableGL	44.54±6.58	45.64±6.25	41.88±6.88
	DeCaf	**51.84±0.88**	**52.29±0.56**	**50.68±0.63**
H2GCN	ERM	49.38±3.44	31.72±1.60	66.59±1.86
	IRM	51.17±8.82	33.57±4.78	62.04±1.45
	EERM	49.04±3.54	30.04±1.72	65.67±1.62
	CIT	54.33±3.90	30.28±3.96	64.78±2.64
	FLOOD	47.18±4.97	29.56±1.13	64.65±0.79
	StableGL	39.99±2.97	37.44±5.21	63.06±4.10
	DeCaf	**55.92±5.20**	**44.96±1.78**	**67.02±1.12**

Table 8: Comparision between methods.

Method	Tailored for graphs	Multiple training envs	Training envs augmentation	Access to test distributions	Test-time training	Complexity
ERM	N/A	N/A	N/A	N/A	N/A	$O\left(nL\bar{d}^2\right)$
IRM	✗	Required	Not Required	Not Required	Not Required	$O\left(nL\bar{d}^2\right)$
REX	✗	Required	Not Required	Not Required	Not Required	$O\left(nL\bar{d}^2\right)$
EERM	✓	Not Required	Required	Not Required	Not Required	$O\left(K\left(nL\bar{d}^2\right)\right)$
CIT	✓	Not Required	Not Required	Not Required	Not Required	$O\left(nL\bar{d}^2\right) + O(K(e+nK)+pn)$
FLOOD	✓	Required	Not Required	Not Required	Required	$O\left((K+2)\left(nL\bar{d}^2\right)\right)$
SR-GNN	✓	Not Required	Not Required	Required	Not Required	$O\left(nL\bar{d}^2 + n^2\right)$
DeCaf	✓	Not Required	Not Required	Not Required	Not Required	$O\left(nL\bar{d}^2\right) + O\left(5n\right)$

K DATASETS AND SETUP

K.1 Hyperparameter Setup

The hidden size of the backbone GNNs of all methods is searched from 8, 16, 32, 64, 128, the number of heads for GAT is searched from 4, 8, and the number of layers is 2. We use Adam as the optimizer with a learning rate of 1e-3 and weight decay of 1e-5. For methods with penalty weights, we searched from different values centered on their default value. For instance, for IRM, the default penalty weight is 1e5, we then conduct our search on 1e2, 1e3, 1e4, 1e5, 1e6, 1e7, 1e8. We follow the default setting for other hyperparameters, such as the number of augmented views. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU with 24GB memory.

K.2 Soft Label Leave-out Setting

In our experiments, if we have 6 classes and the training set has 80% from the first two classes, 10% from the second two classes, and 10% from the last two classes. Then, the validation set owns 10% from the first two classes, 80% from the second two classes, and 10% from the last two classes. Test set owns 10% from the first two classes, 10% from the second two classes, and 80% from the last two classes.

Table 9: Statistics of real-world single graph datasets.

	Cora	Citeseer	Amazon-photo	Coauthor-CS	Squirrel	Roman-empire	Tolokers
# Node	2,708	3,327	7,650	18,333	2,223	22,662	11,758
# Edge	5,278	4,552	119,081	81,894	23,499	32,927	259,500
# Class	7	6	8	15	5	18	2
# Feat	1433	3703	745	6,805	2,089	300	10
Metric	Marco-F1	Marco-F1	Marco-F1	Marco-F1	Marco-F1	Marco-F1	F1 score

Table 10: Statistics of the Facebook-100 dataset.

	Johns Hopkins	Caltech	Amherst	Bingham	Duke	Princeton	WashU
# Node	5,180	769	2,235	10,004	9,895	6,596	7,755
# Edge	373,172	33,312	181,908	725,788	1,012,884	586,640	735,082
Positive rate	43%	53%	36%	40%	39%	37%	38%
	Brandeis	Carnegie	Penn	Brown	Texas	Cornell5,	Yale
# Node	3,898	6,637	41,554	8,600	31,560	18,660	8,578
# Edge	275,134	499,934	2,724,458	769,052	2,439,300	1,581,554	810,900
Positive rate	30%	47%	43%	32%	37%	37%	35%

K.3 Real-word Datasets

We provide the statistics of single-graph datasets in Table 9. Cora [McCallum et al., 2000], Citeseer [Giles et al., 1998], and Coauthor-CS [Shchur et al., 2019] are citation networks. Amazon-photo [McAuley et al., 2015] is a co-purchase network where nodes represent goods for sale on e-commerce websites. Squirrel, Roman-empire, and Tolokers are heterophilous networks created by Platonov [Platonov et al., 2024] et al. Squirrel is a Wikipedia network, Roman-empire is created based on the Roman Empire article from English Wikipedia, and Tolokers is created based on data from the Toloka crowdsourcing platform, where the nodes represent tolokers (workers).

Facebook-100 [Traud et al., 2012] contains 100 social networks collected from universities in the United States. Each node represents a student and the goal is to predict the gender of each student. We provide the statistics of sub-datasets we use in Table 10.

OGB-elliptic [Rozemberczki et al., 2021] is a dynamic financial network dataset that contains in total of 43 graph snapshots from different time steps. Each node represents a Bitcoin transaction, and the goal is to detect illicit transactions. We group all 43 snapshots into 7 timeslots and provide statistics for each timeslot in Table 11.

K.4 Synthetic Datasets

We create three synthetic graphs to simulate the situations where node features and neighborhood representation are dependent or independent of each other. For each graph, we randomly sample n instances of $\mathbf{z}$ with 16 features from a multivariate normal distribution. We generate node features, labels, and adjacency matrices based on the data generation process in assumption 1. By posing different constraints on $\mathcal{M}_x$, $\mathcal{M}_y$, $\mathcal{M}_s$, and $\mathcal{M}_o$, we can control the dependence/ independence between node features and neighborhood representation, and their contributions to the labels. Statistical details of these datasets are shown in Table 12.

h-feat: When $\mathcal{M}_x$ can only "observe" half of the elements of $\mathbf{z}$ and $\mathcal{M}_s, \mathcal{M}_o$ can fully observe $\mathbf{z}$, node features and neighborhood representation are correlated with each other, and each of them can reveal extra information about node label. To do so, we assign $\mathcal{M}_x$ as a 16×8 matrix with its first 8 rows as an identical matrix, and the rest rows are all zeros, such that the second half elements of $\mathbf{z}$ do not participate in the construction of $\mathbf{x}$. We assign $\mathcal{M}_s$ and $\mathcal{M}_o$ to be the same size and value as $\mathcal{M}_y$, which is guaranteed to be a non-trivial transformation of $\mathbf{z}$.

qtr-feat: When $\mathcal{M}_x$ "observe" quarter of the elements of $\mathbf{z}$ and $\mathcal{M}_s, \mathcal{M}_o$ can observe the rest three fourth, node features and neighborhood contributes to the prediction of node labels independently. We assign $\mathcal{M}_x$ as a 16×4 matrix with its first 4 rows as an identical matrix, and the rest rows are all zeros. We assign $\mathcal{M}_s$ and $\mathcal{M}_o$ to be the same size and value as $\mathcal{M}_y$, except with the first 4 rows replaced by all zeros.

Table 11: Statistics of the `OGB-elliptic` dataset.

Time slot	1-6	7-12	13-18	19-24	25-30	31-36	37-43
# Node	28,571	18,525	25,985	14,337	24,878	25,920	29,684
# Edge	33,835	19,613	29,274	15,296	28,223	29,689	33,659
Positive rate	11%	22%	12%	23%	12%	10%	3%

Table 12: Statistics of synthetic datasets.

	`h-feat`	`qtr-feat`	`full-feat`
# Node	8,000	8,000	8,000
# Edge	404,597	1,487,637	35,850
# Class	4	4	4
# Feat	8	4	16

`full-feat`: We create another graph where node features and neighborhoods contribute to the prediction of node labels independently in an alternative way. First, we make $\mathcal{M}_x$ fully observe $\mathbf{z}$ by assigning it as a 16×16 identical matrix; then by assigning $\mathcal{M}_s$ and $\mathcal{M}_o$ as zeros, we create completely random edges. We slightly modify equation 2 to be $\mathbf{y}_i = \mathcal{M}_y(\frac{1}{2}\mathbf{z}_i + \frac{1}{2}\bar{\mathbf{z}}_i) + \mathbf{b}_y$, where $\bar{\mathbf{z}}_i$ is the mean $\mathbf{z}$ embedding of neighboring nodes of node v_i, such that the neighborhood representation can directly affect node labels.

L LIMITATIONS

This paper focuses on homogeneous graphs with limited node and edge types. In the future, we plan to extend the method to heterogeneous graphs with more diverse node relations and neighborhood patterns. This extension will broaden the applications of our method to domains such as social networks, healthcare, and biological networks, where heterogeneity can provide rich information for making predictions.

M BROADER IMPACTS

DeCaf improves the generalizability of the GNN, helping it learn a faithful mapping between inputs and outputs that captures true correlations. This is crucial for critical domains vulnerable to security issues, such as cybersecurity, finance, and healthcare. Learning a robust and generalizable model under potential distribution shifts is essential for these domains.