PH.D. DISSERTATION

Graph Structure as A Double-Edge Sword in Machine Learning

Lu Lin

Department of Computer Science
School of Engineering and Applied Science
University of Virginia

July 1, 2022

Abstract

Graph structure has been a general language describing relational data and interconnected systems. It widely exists in a variety of domains: biological and chemical molecule structures studied in natural science, social and economic networks formed in our daily life, the virtual Internet and physical transportation networks built to connect the world. The ubiquity of such a graph-structured description of our world calls for effective and trustworthy machine learning models that can better make use of and learn to understand information represented in such a structured form.

The graph structure of data brings opportunities as well as challenges to the development of practical machine learning solutions. On one hand, the graph structure introduces informative relational inductive bias which can facilitate learning algorithms to reveal fundamental properties of entities and their interactions; on the other hand, such structure imposes complex dependency relationships among entities which could be used in an undesired way threatening the trustworthiness of machine learning.

The thesis presents our understanding towards the computational questions about the double-edged role of graph structure in machine learning. Specifically, we elaborate the opportunities and the trustworthiness issues brought by the graph structure through two threads of my research: 1) how to utilize the information encoded in graph structure to enhance machine learning, especially when label information is limited; 2) how to mitigate potential pitfalls brought by biased or perturbed structure that threaten the fairness and robustness of machine learning. Our study answers these questions lying at the intersection of machine learning, graph theory and network science. By unleashing the power of graph structure while mitigating the potential pitfalls in machine learning, the outcome of this thesis can be applied to a variety of real-world problems, such as recommendation, user modeling, document understanding, and representation learning.

Acknowledgements

I have been very fortunate to receive support and inspiration from many people during my Ph.D. study. Without them, I could not have gone through this long and rewarding journey, which greatly sharpened my skillset, discovered my potentials and reinvented myself.

I want to sincerely thank my advisor Prof. Hongning Wang, who has been and will always be the most inspiring role model in my academic life. He is a super modest and smart researcher of rock-solid integrity. His vision and insight on research have been my eyes in the junior years when I was struggling in planing my future research. His rigorous, diligent attitude and integrity set a high standard for me to reach. His enthusiasm for research inspired me each time when I was taken over by negative thoughts. I could not have overcome the difficulties and began to enjoy research life without his constant encouragement, patience and countless supports. I would miss the inspiring discussions, the heart-to-heart talks and happy teatimes Prof. Wang have been generously offered.

I would like to thank my dissertation committee members Prof. Aidong Zhang, Prof. Xia Hu, Prof. Jundong Li, and Prof. Teague Henry for their invaluable suggestions on my Ph.D. research and dissertation. Prof. Aidong Zhang's comments broaden the scope of my research and inspire me to view problems from different angles. Prof. Xia Hu and Prof. Jundong Li ask insightful questions from an expert perspective, which can always strike home and drive me to the fundamental issues. I thank Prof. Teague Henry, for sharing his expertise from the network science perspective, which greatly helped me improve and complement my thesis.

I want to thank my collaborators who have been vital supporters for my research. I want to thank Prof. Feng Chen who has been the first mentor in my research journey. Without his encouragement and sharing, I may not have been so determined to pursue Ph.D. degree. I would also like to thank my mentors Jian Qiao and Yang Yang at LinkedIn and Yupeng Gu, Ozan Koyluoglu and Bo Zhao at Pinterest for the mentorship and supports during my internships. I thank all members of our HCDM group for the wonderful research atmosphere, which spawned many interesting discussions and teatime activities. I have been inspired and learned a lot from each of them. I have also worked closely with Lin Gong, Nan Wang and Ethan Blaser. I thank them for the collaborations and their great supports.

I would also like to thank National Science Foundation for the indispensable support to my research and Ph.D. program via grants IIS-1553568 and IIS-1718216.

Special thank to my partner, Jinghui, who has been the sunshine during my dark days. When I think myself not good enough and fear to try, he has always been supportive and thoughtful to help me accept my limitations, stop judge myself, and pursuit the best effort. I do learn enormously from his inner strength and mild characteristics. I become a better person because of him. I love us for our belief in each other as a team.

Finally, I want to thank my parents for their unconditionally support and love. Regardless of ups and downs, they are my most powerful backing.

To myself and my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1 Overview: When Graph Structure Meets Machine Learning

Graph structure is all around us: from fundamental physical interactions to emergent structures such as atoms, molecules, living organisms, societal networks, ecosystems, planetary systems, and many more examples across a wide range of scales in the universe. Not only is the world around us rich in structure, but also our own understanding of the world is highly structured: we think, reason and communicate in terms of relations. A natural way to formalize and represent information in structured form is as a *graph*, which is a data structure describing a collection of entities, represented as *nodes*, and their pairwise relationships, represented as *edges*. The ubiquity of such a graph-structured description of our world calls for the development of effective and trustworthy methods that make use of and learn to understand information represented in this structured form. While we get there, the lessons learned along the way can help us develop better intelligent systems and agents that share a similarly structured understanding of the world as we humans do.

The desire to understand human cognition (e.g., how humans perceive and correlate things) has spawned a variety of scientific disciplines in artificial intelligence. This proposal is inspired by graph theory and network science to study the relations and the collective graph structure of data in the field of machine learning. Machine learning deals with the question of how we can build systems and design algorithms that *learn* from data. The problem of learning is commonly approached by fitting a *model* to data with the goal that this learned model will generalize to new data. *Supervised* machine learning has been proven powerful in many domains, which aims to maximize the likelihood of observing labels given features. However, it heavily relies on well-annotated manual labels, which require domain knowledge and thus are difficult to obtain. *Unsupervised* machine learning does not require label information and aims to discover intrinsic patterns from the data itself. The issue is that unsupervised machine learning is usually inefficient due to the lack of clear and optimal goals for the model to learn.

With the graph structure of data, we can now bridge the gap between supervised and unsupervised machine learning. The graph structure provides explicit, implicit, and global information which can effectively guide the learning process, and the information is freely offered with the generation of graph-structured data. The *explicit* relationships/edges between entities indicate whether two entities should be connected, which can naturally serve as label information in many machine learning tasks, e.g., link prediction. The relationships can also guide the feature aggregation process by introducing relational inductive bias for many machine learning models, e.g., graph convolutional networks. The *implicit* information reflects the clustering property of graphs, which can capture and describe collective patterns of entities to help machine learning models study the behavior of individual entities. The *global* structural property of the graph collects the important essence of the graph to differentiate it from other graphs,

which can be comprehensively summarized by the graph spectrum. This suggests the tremendous opportunities brought by graph structure to machine learning paradigms.

While we take advantage of graph structure to advance machine learning, we cannot ignore potential pitfalls when the graph structure is used in an undesired way. The misuse of graph structure could lead to serious threats to the accountability and trustworthiness of machine learning. Ethical issues arise when a machine learning model is trained on a biased structure, which could output unfair decision making to different groups of users based on their sensitive demographic features, and simply removing sensitive features cannot amend the biased structure. Another concern is regarding security when the graph structure which is used in a machine learning model can be penetrated by an adversary. These issues indicate that introducing graph structure in machine learning could be Pandora opening her box, since biased or perturbed graph structure could greatly mislead the machine learning models. This calls for careful treatment of graph structure when dealing with real-world applications, such as job recommendations or friend recommendations.

In this dissertation, we study the double-edged sword role of graph structure for machine learning, given its information advantage and shortcomings. Based on such fact, we aim to develop machine learning solutions that can model graph structure in an effective and trustworthy way. To achieve this goal, two main challenges arise. One is to discover and harness different types of information encoded in the graph structure, such that machine learning models can be enhanced especially when label information is limited. The other is to understand how biased or perturbed structures threaten the fairness and robustness of machine learning. Our study answers these questions lying at the intersection of machine learning, graph theory, and network science.

2 Challenges and Contributions

To enable effective and trustworthy deployment of machine learning solutions on graph-structured data, we need to address two challenges that stand between machine learning paradigms and non-Euclidean graph structure. Addressing these two challenges consists of two main threads of this dissertation.

2.1 Challenge I: Harnessing Graph Structure in Machine Learning

Modeling and analyzing the graph structure bring enormous opportunities to reveal fundamental properties of entities (e.g., you are whom you connect with) and understand how entities interact with each other (e.g., social influence and contextual effect). Nevertheless, the complex dependency relationships imposed by the graph structure also introduces nontrivial requirements for machine learning to harness real-life graphs (e.g., non-i.i.d. problem instances), especially on rich-content and unlabeled data. The major challenge is to enable machine learning models to effectively discover different types of structural information: the explicit links indicate the dependency between two entities, which could be even more complicated when other data modalities exist such as text; the entities implicitly form groups and clusters reflecting higher-order patterns among entities beyond pairwise relationships; graph structure also globally present essential properties to differentiate the graph from others, which can be summarized by the graph spectrum. Multiple levels of structural information provides a comprehensive profile for the relational-data, and failing to discover and exploit such information will lead to a suboptimal models.

In this thread, we explore possibilities in exploiting the explicit, implicit and global structural information encoded by the graph structure in graph neural networks. Specifically, our contributions can be summarized to answer the following questions.

Research Question 1: *How to advance machine learning on rich-content but unlabeled data using explicit relationship information within data?*
Multimodal contents (e.g., texts and images) are commonly associated with, and manifest, the structure, e.g., users post textual comments when they interact with others. This imposes a higher demand on simultaneously harnessing the structure and content to understand entities even when labels are not available. Our main contribution to address this question is a novel graph neural network, which is discussed in Chapter 2, Section 1. This model is equipped with a channel-aware attention mechanism enabled by edge text content when aggregating information from neighboring nodes. We propose a fresh principle for joint modeling text content and graph structure, and realize the principle via a variational auto-encoder on both text and graph.

Research Question 2: *How to discover and exploit implicit cluster information to further advance machine learning with graph structure?*
Graph structure provides higher-order and global structure patterns, which is beyond the pairwise relationships. Canonical graph convolutional networks cannot differentiate the importance of relationships when aggregating the neighboring states. Our contribution to address this question is a cluster-based graph convolutional network discussed in Chapter 2, Section 2. We argue that the implicit clusters can offer another way to relate nodes, and nodes that are belong to the same cluster should be embedded closer. We realize this principle by uncovering the hierarchical clustering of nodes, and adding a group-level attention mechanism to guide the feature aggregation.

Research Question 3: *How to advance self-supervised learning with graph structural property?*
Self-supervised learning on graphs has gained attentions to capture intrinsic structural patterns by designed supervised signals. Graph contrastive learning is one of the dominating solution to maximize the correspondence between graphs in different augmented views. Our main contribution is to generate augmentation views based on the spectral property of graphs, discussed in Chapter 2, Section 3. The augmentation should inject structural perturbations that do not contain too much redundant information and cannot greatly modify the global property of the graph. We aim to use the spectral change of graph to control the trade-off when generating augmentations.

2.2 Challenge II: Understanding Structural Threats in Machine Learning

Machine learning models have been shown unfair to biased dataset and vulnerable to adversarial examples. These issues threatening the trustworthiness of machine learning becomes even severe when handling relational data where entities are highly correlated: the structure can be biased when the sensitive attributes are involved in the generation of relational data, such that the bias is amplified in learned embeddings when neighboring nodes with similar sensitive attributes are aggregated in the message passing paradigm; an attacker can utilize the vulnerability of the structure to inject hacking nodes or edges, such that a learning model is fooled to make erroneous predictions on target nodes. The existence of bias and perturbation on real-world graph structure posts severe challenges to the trustworthiness of machine learning models.

In this thread, we develop unbiased graph embedding methods to mitigate structural bias, and evaluate the model robustness when the global structural property reflected by the graph spectrum is changed. The contribution of this thread of works are guided by the following research questions.

Research Question 4: *How to prevent the structure from spreading and amplifying bias on entities?*
The network structure driven by homophily and social influence is inevitably affected by sensitive attributes of entities, e.g., people with the same skin color tend to connect, and a loan or job recommender system learned on such structure may favor or disregard on groups. The structure can inherit and even

magnify undesired societal discrimination, which raises fairness issues. Our main contribution is to learn unbiased graph embeddings from biased graph structure, discussed in Chapter 3, Section 1. We study the generation process of attributed graph, and build the connection between the edge distributions of the observed biased graph and a underlying bias-free graph. We then exploit such connection to reweigh the biased training objective, which in expectation achieves the same effect of learning on the underlying bias-free graph. We also propose to regularize the training objective by minimizing the discrepancy between group-level edge distributions on these two graphs.

Research Question 5: *How to understand and evaluate the model robustness when the structure is maliciously altered?*
The network structure exposing dependency information gives malicious attackers more room to break in, e.g., phishing users can deceive recommender systems and worm normal users' trust by strategically making connections with their friends. This raises trustworthy issues and security concerns. Our main contribution is to study the structural robustness in the Fourier domain, discussed in Chapter 3, Section 2. The graph spectral filters are the key in graph convolutional networks, and we propose to directly disrupt the spectral filters by generating structural perturbation that maximize the spectral changes, such that the graph convolutional networks are undermined.

3 Dissertation Structure

To answer the proposed questions which are essential to understand and exploit graph structure to achieve a more effective and trustworthy machine learning, the rest of the thesis is organized as follows. In Chapter 2, we thoroughly study the different types of information encoded in the graph structure, including explicit links, the implicit hierarchical grouping of entities, and global spectral property. We develop practical unsupervised generative models and contrastive learning paradigms to harness the graph structure of unlabeled data. In Chapter 3, we propose principled frameworks to mitigate structural bias and understand structural vulnerability in representation learning. We first cope with the ethical issue by studying the graph generation process, based on which we learn node embeddings from an underlying bias-free graph whose edges are generated without any influence from sensitive attributes. We then study the structural vulnerability in the Fourier domain by designing a structural attack model to directly maximize the spectral distance between the original and perturbed graph. Addressing the proposed research problems can establish a solid foundation to develop machine learning paradigms for rich-content, biased and perturbed graph structure, which widely exist in many real-world domains and applications.

Chapter 2

Harnessing Graph Structure in Unsupervised Machine Learning

Supervised machine learning heavily relies on well-annotated manual labels. However, the cost of label annotation and collection is prohibitive for many research problems in data science which deal with large-scale datasets (e.g., social networks with millions of entities) and problems that rely on domain knowledge (e.g., drug design in biochemistry). Considering a large amount of unlabeled data (e.g., free text and network structure) that can be freely obtained, unsupervised machine learning is studied to uncover intrinsic patterns within data without soliciting label information. The graph structure of data provides enormous information which can be harnessed to advance unsupervised machine learning. Given the ubiquity of graph structure in a broad spectrum of domains, one thread of my dissertation aims to answer the question: *how to discover knowledge from relational but unlabeled structure?*

In this chapter, by thoroughly studying the different types of information encoded in the graph structure, we provide insights on better exploiting the graph structure in unsupervised learning paradigms. In Section 1, we explicitly use the edge as an indicator of similarity between entities, and propose to jointly embed network and topic from user-generated data [50]. This work is based on a simple but intrinsic principle: the contents and connections are generated from entities' hidden essence, which can be modeled as latent variables and learned as representations. In Section 2, we discover a latent hierarchical grouping of entities that commonly exist in network structure: entities sharing common property and neighbors tend to group and small groups can be merged into larger groups in a hierarchical way [100]. This work verifies the informativeness of discovering and preserving implicit structures. In Section 3, we explore a newly emerging self-supervised learning paradigm, a.k.a. contrastive learning, on graphs, and propose a graph augmentation scheme guided by graph spectrum capturing global structural property. Generally speaking, my works study the *generation of unlabeled data* to understand the dependency structure in relational data, and develop practical models for real-world problems such as representation learning, user modeling, and recommendation.

1 Unsupervised Machine Learning with Explicit Structure

We first study the problem of *user modeling* in social networks as a practical showcase of exploiting explicit graph structure for unsupervised machine learning. Essentially, user modeling builds up conceptual representations of users, which help automated systems to better capture users' needs and enhance user experience in such systems [46, 84]. Extensive efforts have proved the value of user representation learning in various real-world applications, such as latent factor models for collaborative filtering [142, 85], topic models for content modeling [182, 115], network embedding models for social link prediction [99, 16], and many more [154, 188].

User representation learning is challenging, and it can never be a straightforward application of existing statistical learning algorithms on user-generated data. First, user-generated data is noisy, incomplete, highly unstructured, and tied with social interactions [165], which imposes serious challenges in modeling such data. For example, in an environment where users are connected, e.g., social network, user-generated data is potentially related, which directly breaks the popularly imposed independent and identically distributed assumptions in most learning solutions [99, 163, 49]. Second, users often participate in various online activities simultaneously, which creates instrumental contextual signals across different modalities. Although oftentimes scattered and sparse, such multi-modal observations reflect users' underlying intents as a whole and call for a holistic modeling approach [88]. Ad-hoc data-driven solutions inevitably isolate the dependency and hence fail to create a comprehensive representation of users. For example, users' social interactions [134, 16] and their generated text data [13, 182, 115] have been extensively studied for user representation learning, but they are largely modeled in isolation. Third, consequently, a unified user representation learning solution is preferred to serve different applications, by taking advantage of data-rich applications to help those data-poor applications.

Even among a few attempts for joint modeling of different types of user-generated data [196, 51], *explicit modeling of dependency* among multiple behavior modalities is still missing. For example, Yang et al. [196] incorporated user-generated text content into network representation learning via joint matrix factorization. In their solution, content modeling is only used as a regularization for network modeling; and thus the learnt model is not in a position to predict unseen text content. Gong and Wang [51] paired the task of sentiment classification with that of social network modeling, and represented each user as a mixture over the instances of these paired tasks. Though text and network are jointly considered, they are only correlated by sharing the same mixing component, without explicitly modeling of the mutual influence between them.

In social psychology and cognitive science, the concept of *user schema* defines the knowledge structure a person holds which organizes categories of information and the relationships among such categories [172]. Putting it into the scenario of user modeling, we naturally interpret the knowledge structure as user representation described by the collection of associated data, such as the set of textual reviews and behavioral logs associated with individual users. The interrelation existing among multiple types of data further motivates us to perform user modeling in a joint manner while the concept of distributed representation learning [8] provides us one possible solution. By constructing a shared latent space, we can embed different modalities of user-generated data in the same low-dimensional space, where the structural dependency among them can be realized by the proximity among different embeddings. The space should be constructed in such a way: 1) the properties of each modality of user-generated data is preserved; 2) the closeness among different modalities of user-generated data can be characterized by the similarity measured in the latent space. For example, connected users in a social network should be closer to each other in this latent space; and by mapping other types of user behavior data into this same space, e.g., text data or behavioral logs, users should be surrounded by their own generated data.

To realize this new perspective of user representation learning, we exploit two most widely available and representative forms of user-generated data, i.e., text content and social interactions. We develop a probabilistic generative model to integrate user modeling with content and network embedding. Due to the unstructured nature of text, we appeal to statistical topic models to model user-generated text content [13, 182], with a goal to capture the underlying semantics. We define a topic as a probability distribution over a fixed vocabulary [13]. We embed both users and topics to the same low-dimensional space to capture of their mutual dependency. On one hand, a user's affinity to a topic is characterized by his/her proximity to the topic's embedding in this latent space, which is utilized to generate each text document of the user. On the other hand, the affinity between users is directly modeled by the proximity between users' embeddings, which are utilized to generate the corresponding social network connections.

In this latent space, the two modalities of user-generated data are correlated explicitly, indicated by the user's topical preferences. The user representation is obtained by posterior inference of those embedding vectors over a set of training data, via variational Bayesian.

1.1 Related Work

In order to learn effective user representations, a lot efforts have been devoted to modeling diverse modalities of user-generated data: 1) in an isolated manner, i.e., focusing on one modality of user-generated data such as text reviews; 2) in a joint manner, i.e., utilizing multiple types of user data. Our proposed solution falls into the second one as it learns representations from both network structure and text content by capturing the dependency between the two modalities in the latent topic space.

When performing user representation learning in an isolated way, much attention has been paid on exploring user-user interactions to learn users' distributed representations, which are essential for better understanding users' interactive preferences in social network analysis. Inspired from word embedding techniques [120], random walk models are exploited to generate random paths over a network to learn dense, continuous and low-dimensional representations of users [134, 166, 52]. Matrix factorization technique is also commonly used to learn user embeddings [128, 185], as learning a low-rank space for an adjacency matrix representing the network naturally fits the need of learning low-rank user/node embeddings. For instance, Tang and Liu [167] factorize an input network's modularity matrix and use discriminative training to extract representative dimensions for learning user representation.

In parallel, the user-generated text data is utilized to understand users' emphasis on specific entities or aspects. Topic models [13, 61] serve as a building block for statistical modeling of text data. Typical solutions model individual users as a bag of topics [146], which govern the generation of associated text documents. Wang and Blei [182] combine topic modeling with collaborative filtering to estimate topical user representations with additional observations from user-item ratings. Wang et al. [183] use topic modeling to estimate users' detailed aspect-level preferences from their opinionated review content. Lin et al. [103] learn users' personalized topical compositions to differentiate user's subjectivity from item's intrinsic property in the review documents. Implicit preferences of each user as well as the properties of each product is uncovered by mapping users and items into a shared topic space [115]. Some recent works use deep neural networks to obtain user embedding from their generated text data [164, 27].

Although most previous works studied social networks and user-generated text content in isolation, little attention has been paid in combining the two sources for better user modeling. Earlier work [118] regularizes a statistical topic model with a harmonic regularizer defined on the network structure. Yang et al. [196] incorporate text features of users into network representation learning via joint matrix factorization. Gong and Wang [51] pair tasks of opinionated content modeling and network structure modeling in a group-wise fashion, and model each user as a mixture over the tasks. Though both text and network are utilized for user modeling in the aforementioned works, explicit modeling of dependence among different modalities is still missing. Archarya et al. [2] explore the dependency among documents and network but on a per-community basis instead of a per-user basis. Our work proposes a holistic view to model users' social preferences and topical interests jointly, thus to provide a more general understanding of user intents from multiple perspectives.

1.2 Method: Joint Network Embedding and Topic Embedding

To interrelate different modalities of user-generated data for user representation learning, we propose to perform joint network embedding and topic embedding. In this section, we first provide the details of our

probabilistic generative model, JNET, which imposes a complete generative process over user-generated social interactions and text data in each individual user. Then we describe our variational Bayesian based Expectation Maximization algorithm, which retrieves the learnt user representation from a given corpus.

1.2.1 Model Specification We denote a collection of U users as $\mathcal{U} = \{u_1, u_2, ...u_U\}$, in which each user u_i is associated with a set of documents $\mathcal{D}_i = \{x_{i,d}\}_{d=1}^{D_i}$. Each document is represented as a bag of words $x_d = \{w_1, w_2, .., w_N\}$, where each w_n is chosen from a vocabulary of size V. Each user is also associated with a set of social connections denoted as $\mathcal{E}_i = \{e_{ij}\}_{j \neq i}^{U}$, where $e_{ij} = 1$ indicates user u_i and u_j are connected in the network; otherwise, $e_{ij} = 0$.

We represent each user as a real-valued continuous vector $u_i \in \mathbb{R}^M$ in a low-dimensional space. And we seek to impose a joint distribution over the observations in each user's associated text documents and social interactions, so as to capture the underlying structural dependency between these two types of data. Based on our assumption that both types of users-generated data are governed by the same underlying user intent, we explicitly model the joint distribution as $p(\mathcal{D}_i, \mathcal{E}_i) = \int p(\mathcal{D}_i, \mathcal{E}_i, u_i) \mathrm{d}u_i$, which can be further decomposed into $p(\mathcal{D}_i, \mathcal{E}_i, u_i) = p(\mathcal{D}_i|\mathcal{E}_i, u_i)p(\mathcal{E}_i|u_i)p(u_i)$. We assume given the user representation u_i, the generation of text documents in $\mathcal{D}_i$ is independent from the generation of social interactions in $\mathcal{E}_i$, i.e., $p(\mathcal{D}_i|\mathcal{E}_i, u_i) = p(\mathcal{D}_i|u_i)$. As a result, the modeling of joint probability over a user's observational data with his/her latent representation can be decomposed into three related modeling tasks: 1) $p(\mathcal{D}_i|u_i)$ for content modeling, 2) $p(\mathcal{E}_i|u_i)$ for social connection modeling, and 3) $p(u_i)$ for user embedding modeling.

We appeal to topic models [13, 61] due to their effectiveness shown in existing empirical studies for content modeling. The concept of user schema inspires us to embed both users and topics to the same latent space in order to capture the dependency between them. By projecting a user's embedding vector to topic embedding vectors, we can easily measure affinity between a user and a topic, and thus capture users' topical preferences. It also allows us to capture the topical variance in documents from the same user and establish a valid predictive distribution of his/her documents.

Formally, we assume there are in total K topics underlying the corpus with each represented as an embedding vector $\phi_k \in \mathbb{R}^M$ in the same latent space; denote $\Phi \in \mathbb{R}^{K \times M}$ as the matrix of topic embeddings, which facilitate our representation of each user's affinity towards different topics: $\Phi \cdot u_i$, which reflects user u_i's topical preferences, and serves as the prior of topic distribution in each text document from him/her. Specifically, denote the document-level topic vector as $\theta_{id} \in \mathbb{R}^K$, we have $\theta_{id} \sim \mathcal{N}(\Phi \cdot u_i, \tau^{-1} I)$, where τ characterizes the uncertainty when user u_i is choosing topics from his/her global topic preferences for each single document. By projecting the document-level topic vector θ_{id} into a probability simplex, we obtain the topic distribution for document $x_{i,d}$: $\pi_{idk} = \text{softmax}(\theta_{idk}) = \exp(\theta_{idk}) / \sum_{l=1}^{K} \exp(\theta_{idl})$, from which we sample a topic indicator $z_{idn} \in \{1, ..., K\}$ for each word w_{idn} in $x_{i,d}$ by $z_{idn} \sim \text{Multi}(\pi_{idk})$. As in conventional topic models, each topic k is also associated with a multinomial distribution β_k over a fixed vocabulary, and each word w_{idn} is then drawn from the respective word distribution indicated by corresponding topic assignment, i.e., $w_{idn} \sim p(w|\beta_{z_{idn}})$. Putting all pieces together, the task of content modeling for each user can be summarized as $p(\mathcal{D}_i|u_i) = \prod_{d=1}^{D_i} p(\theta_{id}|u_i, \Phi, \tau) \prod_{n=1}^{N} p(z_{idn}|\theta_{id}) p(w_{idn}|z_{idn}, \beta)$.

The key in modeling social connections is to understand the closeness among users. As we represent users with a real-valued continuous vector, this can be easily measured by the vector inner product in the learnt low-dimensional space. Define the underlying affinity between a pair of users u_i and u_j as δ_{ij}, we assume $\mathbb{E}[\delta_{ij}] = u_i^{\mathsf{T}} u_j$. To capture uncertainty of the affinity between different pairs of users, we further assume δ_{ij} is drawn from a Gaussian distribution centered at the measured closeness,

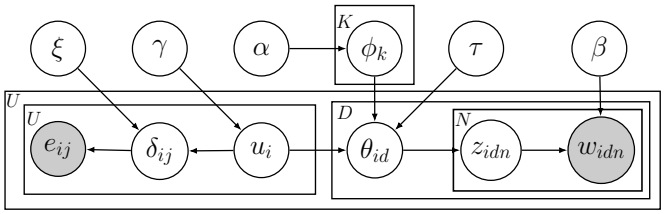


Fig. 1: Graphical model representation of JNET. The upper plate indexed by K denotes the learnt topic embeddings. The outer plate indexed by U denotes distinct users in the collection. The inner plates indexed by U and D denote each user's social connections and text documents respectively. The inner plate indexed by N denotes the word content in one text document.

$\delta_{ij} \sim \mathcal{N}(u_i^\mathsf{T} u_j, \xi^2)$, where ξ characterizes the concentration of this distribution. The observed social connection e_{ij} between user u_i and u_j is then assumed as a realization of this underlying user affinity: $e_{ij} \sim$ Bernoulli(logistic(δ_{ij})) where logistic$(\delta_{ij}) = 1/(1 + \exp(-\delta_{ij}))$. As a result, the task of social connection modeling can be achieved by $p(\mathcal{E}_i|u_i) = \prod_{j \neq i}^{U} p(e_{ij}|\delta_{ij}) p(\delta_{ij}|u_i, u_j)$.

We do not have any specific constraint on the form of latent user embedding vectors $\{u_i\}_{i=1}^{U}$ and topic embedding $\{\phi_k\}_{k=1}^{K}$, as long as they are in a M-dimensional space. For simplicity, we assume they are drawn from isotropic Gaussian distributions respectively, i.e., $u_i \sim \mathcal{N}(\mathbf{0}, \gamma^{-1}\boldsymbol{I})$, where γ measures the concentration of different users' embedding vectors, and $\phi_k \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\boldsymbol{I})$. Other types of prior distribution can also be introduced, if one has more knowledge about the user and topic embeddings, such as sparsity or a particular geometric shape. But it is generally preferred to have conjugate priors, so as to simplify later posterior inference steps.

Putting these components together, the generative process of our solution can be described as follows:

- For each topic ϕ_k:
  - Draw its topic compact representation $\phi_k \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\boldsymbol{I})$
- For each user u_i:
  - Draw its user compact representation $u_i \sim \mathcal{N}(\mathbf{0}, \gamma^{-1}\boldsymbol{I})$
  - For every other user u_j:
    - Draw affinity δ_{ij} between u_i and u_j, $\delta_{ij} \sim \mathcal{N}(u_i^\mathsf{T} u_j, \xi^2)$
    - Draw interaction e_{ij} between u_i and u_j, $e_{ij} \sim$ Bernoulli(logistic(δ_{ij}))
- For each document of user u_i:
  - Draw the user-document topic preference vector $\theta_{id} \sim \mathcal{N}(\Phi \cdot u_i, \tau^{-1}\boldsymbol{I})$
  - For each word w_{idn}:
    - Draw topic assignment $z_{idn} \sim$ Multi(softmax(θ_{id}))
    - Draw word $w_{idn} \sim$ Multi$(\beta_{z_{idn}})$

We make two explicit assumptions here: 1) the dimensionality M of the compact representation of topics and users is predefined and fixed; 2) the word distributions under topics are parameterized by a $K \times V$ matrix β where $\beta_{kv} = p(w^v|z^k)$ over a fixed vocabulary of size V. The generative model captures the interrelation between multiple modalities of user-generated data for user representation learning. In essence, we are performing a **J**oint **N**etwork **E**mbedding and **T**opic Embedding, thus, we name the resulting model as JNET in short.

1.2.2 Variational Bayesian Inference The compact user representations can be obtained via posterior inference over the latent variables on a given set of data. However, posterior inference is not analytically tractable in JNET due to the coupling among latent variables, i.e., user-user affinity δ, user embedding u, topic embedding Φ, document-level topic proportion θ and word-level topic assignment z. We appeal to a mean-field variational method to approximate the posterior distributions, and further utilize Taylor expansion [12] to address the difficulty introduced by non-conjugate logistic-normal priors.

We begin by postulating a factorized distribution:

$$q(\Phi, U, \Delta, \Theta, Z) = \prod_{k=1}^{K} q(\phi_k) \prod_{i=1}^{U} q(u_i) \Big[\prod_{j=1, j\neq i}^{U} q(\delta_{ij}) \prod_{d=1}^{D} q(\theta_{id}) \prod_{n=1}^{N} q(z_{idn}) \Big]$$

where the factors have the following parametric forms:

$$\begin{aligned}
&q(\phi_k) = \mathcal{N}(\phi_k | \mu^{(\phi_k)}, \Sigma^{(\phi_k)}), q(u_i) = \mathcal{N}(u_i | \mu^{(u_i)}, \Sigma^{(u_i)}), \\
&q(\delta_{ij}) = \mathcal{N}(\delta_{ij} | \mu^{(\delta_{ij})}, {\sigma^{(\delta_{ij})}}^2), q(\theta_{id}) = \mathcal{N}(\theta_{id} | \mu^{(\theta_{id})}, \Sigma^{(\theta_{id})}), \\
&q(z_{idn}) = \text{Mult}(z_{idn} | \eta_{idn})
\end{aligned}$$

Because the topic proportion vector θ_{id} is inferred in each document, it is not necessary to estimate a full covariance matrix for it [12]. Hence, in its variational distribution, we only estimate the diagonal variance parameters.

Variational algorithms aim to minimize the KL divergence from the approximated posterior distribution q to the true posterior distribution p. It is equivalent to tightening the evidence lower bound (ELBO) by Jensen's inequality [13]:

$$\log p(\boldsymbol{w}, \boldsymbol{e} | \alpha, \beta, \gamma, \tau) \geq \mathbb{E}_q[\log p(U, \Theta, Z, \Phi, \Delta, \boldsymbol{w}, \boldsymbol{e} | \alpha, \beta, \gamma, \tau)] - \mathbb{E}_q[\log q(U, \Theta, Z, \Phi, \Delta)] \quad (1)$$

where the expectation is taken with respect to the factorized variaitonal distribution of the latent variables $q(\Phi, U, \Delta, \Theta, Z)$. Let $\mathcal{L}(q)$ denote the right-hand side of Eq (1), the first step of maximizing this lower bound is to derive the analytic form of posterior expectations required in $\mathcal{L}(q)$. Thanks to the conjugate priors introduced on $\{u_i\}_{i=1}^{U}$ and $\Phi = \{\phi_k\}_{k=1}^{K}$, the expectations related to these latent variables have closed form solutions, while due to non-conjugate logistic-normal priors, we use Taylor expansions to approximate the expectations related to θ_{id}, δ_{ij}. Next we describe the detailed inference procedure for each latent variable.

• **Estimate Topic Embedding.** For each topic k, we relate the terms associated with $q(\phi_k | \mu^{(\phi_k)}, \Sigma^{(\phi_k)})$ in Eq (1) and take maximization w.r.t. $\mu^{(\phi_k)}$ and $\Sigma^{(\phi_k)}$. Closed form estimations of $\mu^{(\phi_k)}, \Sigma^{(\phi_k)}$ exist,

$$\begin{aligned}
\mu^{(\phi_k)} &= \tau \Sigma^{(\phi_k)} \sum\nolimits_{i=1}^{U} \sum\nolimits_{d=1}^{D_i} \mu_k^{(\theta_{id})} \mu^{(u_i)} \\
\Sigma^{(\phi_k)} &= \big[\alpha \boldsymbol{I} + \tau \sum\nolimits_{i=1}^{U} \sum\nolimits_{d=1}^{D_i} (\Sigma^{(u_i)} + \mu^{(u_i)} {\mu^{(u_i)}}^{\mathsf{T}}) \big]^{-1}
\end{aligned} \quad (2)$$

The estimation of $\Sigma^{(\phi_k)}$ is not related to a specific topic k, because we impose an isotropic Gaussian prior for all $\{\phi_k\}_{k=1}^{K}$ in JNET. It suggests that the correlations between different topic embedding dimensions are homogeneous across topics. Interestingly, we can notice that the posterior covariance $\Sigma^{(\phi_k)}$ of topic embeddings is closely related to user embeddings, which indicates direct dependency from network structure to text content.

• **Estimate User Embedding.** For each user i, we relate the terms associated with $q(u_i | \mu^{(u_i)}, \Sigma^{(u_i)})$ in Eq (1) and maximize it with respect to $\mu^{(u_i)}, \Sigma^{(u_i)}$. Closed form estimations can also be achieved for

these two parameters as follows:

$$\begin{aligned}\mu^{(u_i)} =& \Sigma^{(u_i)}\big(\tau \sum\nolimits_{d=1}^{D_i} \sum\nolimits_{k=1}^{K} \mu_k^{(\theta_{id})} \mu^{(\phi_k)} + \sum\nolimits_{j\neq i}^{U} \xi^{-2} \mu^{(\delta_{ij})} \mu^{(u_j)}\big) \\ \Sigma^{(u_i)} =& \gamma \boldsymbol{I} + \tau D_i \sum\nolimits_{k=1}^{K} (\Sigma^{(\phi_k)} + \mu^{(\phi_k)} \mu^{(\phi_k)\mathsf{T}}) \\ &+ \sum\nolimits_{j\neq i}^{U} \xi^{-2} (\Sigma^{(u_j)} + \mu^{(u_j)} \mu^{(u_j)\mathsf{T}})\end{aligned} \tag{3}$$

The effect of joint content modeling and network modeling for user representation learning is clearly depicted in this posterior estimation of user embedding vectors. The updates of $\mu^{(u_i)}$ and $\Sigma^{(u_i)}$ come from two types of influence: the text content and social interactions of the current user. For example, the posterior mode estimation of user embedding vector u_i is a weighted average over the topic vectors that this user has used in his/her past text documents and the user vectors from his/her friends. And the weights measure his/her affinity to those topics and users in each specific observation. The updates exactly reflect the formation of "user schema" in social psychology from two perspectives: both modalities of user-generated data shape user embeddings, while the structural dependency between them is reflected in this unified user representation.

• **Estimate Per-document Topic Proportion Vector.** Similar procedures as above can be taken to estimate $\mu^{(\theta_{id})}$ and $\Sigma^{(\theta_{id})}$. Due to the lack of conjugate prior for logistic Normal distributions, we apply Taylor expansion and introduce an additional free variational parameter ζ in each document. Because there is no closed form solution for the resulting optimization problem, we use gradient ascent to optimize $\mu^{(\theta_{id})}$ and $\Sigma^{(\theta_{id})}$ with the following gradients,

$$\begin{aligned}\partial L/\partial \mu_k^{(\theta_{id})} =& -\tau \mu_k^{(\theta_{id})} + \tau \mu^{(\phi_k)\mathsf{T}} \mu^{(u_i)} + \sum\nolimits_{n=1}^{N} \left[\eta_{idnk} - \zeta^{-1} \exp(\mu_k^{(\theta_{id})} + \Sigma_{kk}^{(\theta_{id})}/2)\right] \\ \partial L/\partial \Sigma_{kk}^{(\theta_{id})} =& -\tau - N \exp(\mu_k^{(\theta_{id})} + \Sigma_{kk}^{(\theta_{id})}/2)/\zeta + 1/\Sigma_{kk}^{(\theta_{id})}\end{aligned} \tag{4}$$

where $\zeta = \sum_{k=1}^{K} \exp(\mu_k^{(\theta_{id})} + \Sigma_{kk}^{(\theta_{id})}/2)$. Since only the diagonal elements in $\Sigma_{id}^{(\theta)}$ are statistically meaningful (i.e., variance), we simply set its off-diagonal elements to zero in gradient update. The gradient function suggests that the document-level topic proportion vector should align with the corresponding compact user representation and topic representation. Although no closed form estimations of $\mu^{(\theta_{id})}$ and $\Sigma^{(\theta_{id})}$ exist, the expected property of $\mu^{(\theta_{id})}$ is clearly reflected: the proportion of each topic in document $x_{i,d}$ should align with this user's preference on this topic (i.e., affinity in the embedding space) and the topic assignment in document content. And the variance is introduced by the uncertainty of per-word topic choice and the intrinsic uncertainty of a user's affinity with a topic.

• **Estimate User Affinity.** Similar approach can be applied here to estimate $\mu^{(\delta_{ij})}$ and ${\sigma^{(\delta_{ij})}}^2$ which govern the latent user affinity. Again, gradient ascent is utilized to optimize $\mu^{(\delta_{ij})}$ and $\Sigma^{(\theta_{id})}$,

$$\begin{aligned}\partial L/\partial \mu^{(\delta_{ij})} &= e_{ij} - \varepsilon^{-1} \exp\left(\mu^{(\delta_{ij})} + {\sigma^{(\delta_{ij})}}^2/2\right) - \xi^{-2}(\mu^{(\delta_{ij})} - \mu^{(u_i)\mathsf{T}} \mu^{(u_j)}) \\ \partial L/\partial \sigma^{(\delta_{ij})} &= -\varepsilon^{-1} \sigma^{(\delta_{ij})} \exp\left(\mu^{(\delta_{ij})} + {\sigma^{(\delta_{ij})}}^2/2\right) - \xi^{-2} \sigma^{(\delta_{ij})} + 1/\sigma^{(\delta_{ij})}\end{aligned}$$

The gradient functions suggest that the latent affinity between a pair of users is closely related with their observed connectivity and their closeness in the embedding space.

• **Estimate Word Topic Assignment.** The topic assignment z_{idn} for each word w_{idn} in document $x_{i,d}$ can be estimated by,

$$\eta_{idnk} \propto \exp\{\mu_k^{(\theta_{id})} + \sum\nolimits_{v=1}^{V} w_{idnv} \log \beta_{kv}\}$$

We execute the above variational inference procedures in an alternative fashion until the lower bound $\mathcal{L}(q)$ defined in Eq (1) converges. The variational inference algorithm postulates strong independence structures between the variational parameters, allowing straightforward **parallel computing**. Since the variational parameters can be grouped by documents: $\mu^{(\theta_{id})}$, $\Sigma^{(\theta_{id})}$ and η, by topics: $\mu^{(\phi_k)}$ and $\Sigma^{(\phi_k)}$, and by users: $\mu^{(u_i)}$, $\Sigma^{(u_i)}$, $\mu^{(\delta_{ij})}$ and ${\sigma^{(\delta_{ij})}}^2$, we perform alternative update in parallel to improve computational efficiency: for example, we fix topic-level parameters and user-level parameters, and distribute the documents across different machines to estimate their own $\mu^{(\theta_{id})}$, $\Sigma^{(\theta_{id})}$ and η in parallel for large collections of user-generated data.

1.2.3 Parameter Estimation When performing the variational inference described above, we have assumed the knowledge of model parameters $\alpha, \gamma, \tau, \xi$ and β. Based on the inferred posterior distribution of latent variables in JNET, these model parameters can be readily estimated by the Expectation-Maximization (EM) algorithm. The most important model parameters are priors for user embedding γ and topic embedding α, and word-topic distribution β. As ξ and τ serve as the variance for user affinity δ_{ij} and document topic proportion vector θ_{id}, and we have large amount of observations in text documents and social connections across all users, our model is less sensitive to their settings. Therefore, we estimate ξ and τ less frequently than α, γ and β.

By taking the gradient of $\mathcal{L}(q)$ in Eq (1) with respect to α and γ, and set the resulting gradient to 0, we get the closed form estimations of α and γ as follows:

$$\alpha = \frac{KM}{\sum_{k=1}^{K}[\sum_{m=1}^{M}\Sigma_{mm}^{(\phi_k)} + \mu^{(\phi_k)^\mathsf{T}}\mu^{(\phi_k)}]}, \gamma = \frac{UM}{\sum_{i=1}^{U}[\sum_{m=1}^{M}\Sigma_{mm}^{(u_i)} + \mu^{(u_i)^\mathsf{T}}\mu^{(u_i)}]}$$

And the closed form estimation for word-topic distribution β can be achieved by,

$$\beta_{kv} \propto \sum_{i=1}^{U}\sum_{d=1}^{D_i}\sum_{n=1}^{N} w_{idnv}\eta_{idnv}$$

where w_{idnv} indicates the nth word in u_i's dth document is the vth term in the vocabulary. The estimation for ξ and τ is omitted for space limit, but they can be easily derived based on Eq (1).

The resulting EM algorithm consists of E-step and M-step. In E-step, the variational parameters are inferred based the procedures described in Section 1.2.2; and in M-step, the model parameters are estimated based on collected sufficient statistics from E-step. These two steps are repeated until the lower bound $\mathcal{L}(q)$ converges over all training data.

Inferring the latent variables with each user and each topic are computationally cheap. By Eq (2), updating the variables for each topic imposes a complexity of $\mathcal{O}\big(KM^2|D|\big)$, where K is the total number of topics, M is the latent dimension, $|D|$ is the total number of documents. By Eq (3), updating the variables for each user imposes a complexity of $\mathcal{O}(M^2U^2)$ where U is the total number of users. Estimating the latent variables for the per-document topic proportion imposes a complexity of $\mathcal{O}(|D|K(\bar{N}+M))$ by Eq (4), where $\bar{N}$ is the average document length. And updating variables for each pair of user affinity takes constant time while there are U^2 affinity variables. With the consideration of the total number of users and topics, the overall complexity for the proposed algorithm is $\mathcal{O}(KM^2|D| + M^2U^2)$.

1.3 Experiments

We evaluated the proposed model on large collections of Yelp reviews and StackOverflow forum discussions, together with their user network structures. Qualitative analysis demonstrates the descriptive

power of JNET through direct mapping of user and topic embeddings into a 2-D space. The explicit modeling of dependency among user-generated data confirms the effectiveness of JNET, as indicated by the model's predictive power in recovering missing links and modeling unseen documents. The learnt user representation also enables accurate content recommendation to users.

• **Datasets.** We employed two large publicly available user-generated text datasets together with the associated user networks: 1) **Yelp**, collected from Yelp dataset challenge [1], consists of 187,737 Yelp restaurant reviews generated by 10,830 users. The Yelp dataset provides user friendship imported from their Facebook friend connections. Among the whole set of users, 10,194 of them have friends with an average of 10.65 friends per user. 2) **StackOverflow**, collected from Stackoverflow.com [2], consists of 244,360 forum discussion posts generated by 10,808 users. While there is no explicit network structure in StackOverflow dataset, we utilized the "*reply-to*" information in the discussion threads to build a user network, because this relation suggests implicit social connections among users based on their expertise and technical topic interest. We ended up with 10,041 connected users, with an average of 5.55 connections per user. We selected 5,000 unigram and bigram text features based on Document Frequency (DF) in both datasets. We randomly split the data for 5-fold cross validation in all the reported experiments.

• **Baselines.** We compared the proposed JNET model against a rich set of user representation learning methods, including topic modeling based solutions, the network embedding methods, and models performing joint modeling of text and network. 1) **Latent Dirichlet Allocation (LDA)** [13] generates the topic distribution in documents across different users, and the user presentation is constructed by averaging the posterior topic proportion of documents associated with a user. 2) **Relational Topic Model (RTM)** [25] explicitly models the connection between two documents and we constructed a user-level network by concatenating all documents of one user in this baseline. 3) **Hidden Factors and Hidden Topics (HFT)** [115] combines latent rating dimensions of users with latent review topics for user modeling. Users' "*upvote*" toward a question is utilized as a proxy of rating in StackOverflow. 4) **Collaborative Topic Regression (CTR)** [182] combines collaborative filtering with topic modeling to explain the observed text content and ratings. 5) **DeepWalk (DW)** [134] takes truncated random walks as input to learn social representations of vertices in the network. 6) **Text-Associated DeepWalk (TADW)** [196] further incorporates text content of vertices into network representation learning under the framework of joint matrix factorization.

• **Parameter Settings.** We set the latent dimensions of user and topic embeddings to 10 in both JNET and baselines as larger dimension gives limited performance improvement but slows down all models considerably. As we tuned the topic size from 10 to 100, we found the learnt topics are most representative and meaningful at around 40 topics. Hence, we set topic number to 40 in the reported experiments. The maximum number of iteration in our EM algorithm is set to 100. Both the source codes and data are available online [3].

1.3.1 The Learnt User Representations

We first study the quality of the learnt user representations from JNET. The learnt user embeddings are mapped to a 2-D space using the t-SNE algorithm and is visualized in Figure 2. For illustration purpose, we simply assign each user to the topic that he/she is closest to, i.e., $\arg\max_k(\phi_k \cdot u_i)$ and we mark users sharing the same interested topic with the same color. We also plot the most representative words of each topic learnt from JNET (i.e., $\arg\max_w p(w|\beta_z)$), with the same color of the corresponding set of users.

[1] Yelp dataset challenge. http://www.yelp.com/dataset_challenge
[2] StackOverflow. http://stackoverflow.com
[3] JNET. https://github.com/Linda-sunshine/JNET.

Fig. 2: Visualizing learned user embedding and topics on Yelp (LEFT) and StackOverflow (RIGHT).

As we can find from the visualization of StackOverflow, users of similar interests are clearly clustered in the 2-D space, which indicates the descriptive power of our learnt user vectors. Meanwhile, we can easily identify the theme of each learnt topic, such as C++ (in light green circle), SQL (in dark purple circle) and java (in light blue circle). It is also interesting to find correlations among the users and topics by looking into their distances. The users in dark green are mainly interested in website development, thus are far away from the users who are interested in C++ (in light green). The users in orange care more about the network communication and they are overlapped with other clusters of users focusing on SQL (in dark purple) and C++ (in light green) as network communication is an important component among different programming languages. Similar observations can also be found on Yelp dataset.

1.3.2 Document Modeling In order to verify the predictive power of the proposed model, we first evaluated the generalization quality of JNET on the document modeling task. We compared all the topic model based solutions by their *perplexity* on a held-out test set. Formally, the perplexity for a set of held-out documents is calculated as follows [13]:

$$perplexity(D_{test}) = \exp\big(-\frac{\sum_{d\in D_{test}} \log p(\boldsymbol{w}_d)}{\sum_{d\in D_{test}} |d|}\big)$$

where $p(\boldsymbol{w}_d)$ is the likelihood of each held-out document given by a trained model. A lower perplexity indicates better generalization quality of a model.

Figure 3 reports the mean and variance of the perplexity for each model with 5-fold cross validation over different topic sizes. JNET achieved the best predictive power on the hold-out dataset, especially when an appropriate topic size is assigned. RTM achieved comparative performance as it utilizes the connectivity information among users, but it is limited by not being able to capture the variance within each user's different documents. The other baselines do not explicitly model network data, i.e., LDA, HFT and CTR, and therefore suffer in their performance.

A good joint modeling of network structure and text content should complement each other to facilitate a more effective user representation learning. Hence, we expect a good model to learn reasonable representations on users lacking text information, a.k.a., cold-start users, by utilizing network structure. We randomly selected 200 users and held out all their text content for testing. Regarding to the number of social connections each testing user has in training data, we further consider three different sets of users, and name them as light, medium and heavy users, to give a finer analysis with respect to the degree of connectivity in cold-start setting. The threshold for categorizing different sets of users is based on the statistics of each dataset; and each group contains 200 users. In particular, we selected 5 and 20 as the

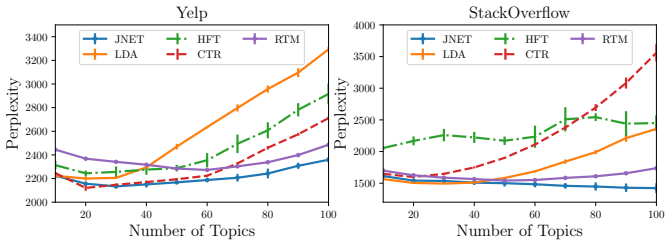


Fig. 3: Perplexity comparison on Yelp and StackOverflow.

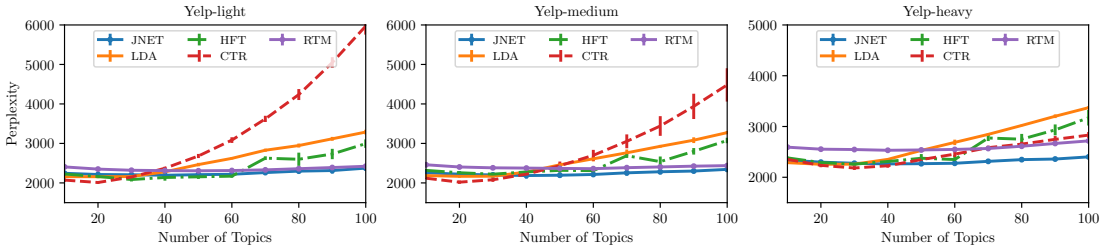


Fig. 4: Comparison of perplexity in cold-start users on Yelp.

connectivity threshold for Yelp, 5 and 15 as the threshold for StackOverflow respectively. That is, in Yelp, light users have fewer than 5 friends, medium users have more than 5 friends while fewer than 20 friends and heavy users have more than 20 friends. We compared JNET against four baselines, i.e., LDA, HFT, RTM, CTR for evaluation purpose. We reported the perplexity on the held-out test documents regarding to the three sets of users, in Figure 4.

As we can observe in Figure 4, JNET performed consistently better on the testing documents for the three different sets of unseen users on Yelp dataset, which indicates the advantage of utilizing network information in addressing cold-start content prediction issue. The benefit of network is further verified across different sets of users as heavily connected users can achieve better performance improvement compared with text only user representation model, i.e., LDA. Similar conclusion is obtained for StackOverflow dataset, while we neglect it due to the space limit.

1.3.3 Link Prediction The predictive power of JNET is not only reflected in unseen documents, but also in missing links. In the task of link prediction, the key component is to infer the similarity between users. We split the observed social connections into 5 folds. Each time, we held out one fold of edges for testing and utilized the rest for model training, together with users' text content. In order to construct a valid set of ranking candidates for each testing user, we randomly injected irrelevant users (non-friends) for evaluation purpose. And the number of irrelevant users is proportional to the number of connections a testing user has, i.e., $t \times$ number of social connections. We rank users based on the cosine similarity between their embedding vectors. Normalized discounted cumulative gain (NDCG) and mean average precision (MAP) are used to measure the quality of ranking. We started with the ratio between irrelevant users and relevant users being $t = 2$ and increased the ratio to $t = 8$ to make the task more challenging to further verify the effectiveness of the learnt user representations.

To compare the prediction performance, we tested five baselines, i.e., LDA, HFT, RTM, DW and TADW. We reported the NDCG and MAP for the two datasets in Figure 5. It is clear JNET achieved encouraging performance on both datasets, which indicates effective user representations are learnt to recover

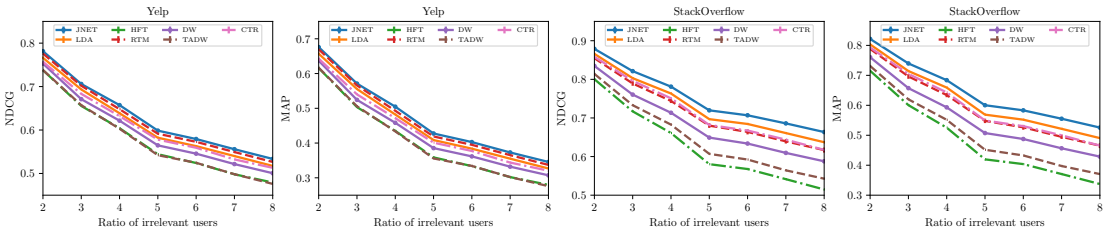


Fig. 5: The performance comparison of link suggestion on Yelp and StackOverflow.

Table 1: The performance comparison of link prediction for cold-start users on StackOvewrflow.

Models	Light			Medium			Heavy		
	Ratio=2 NDCG/MAP	Ratio=4 NDCG/MAP	Ratio=6 NDCG/MAP	Ratio=2 NDCG/MAP	Ratio=4 NDCG/MAP	Ratio=6 NDCG/MAP	Ratio=2 NDCG/MAP	Ratio=4 NDCG/MAP	Ratio=6 NDCG/MAP
LDA	0.786/0.648	0.664/0.477	0.632/0.431	0.774/0.597	0.677/0.451	0.612/0.364	0.818/0.581	0.745/0.443	0.697/0.366
HFT	0.666/0.493	0.543/0.333	0.483/0.259	0.671/0.461	0.562/0.313	0.492/0.226	0.682/0.389	0.591/0.250	0.532/0.179
RTM	0.777/0.642	0.688/0.514	0.627/0.433	0.801/0.638	0.709/0.495	0.654/0.419	0.837/0.624	0.760/0.481	0.711/0.399
TADW	0.695/0.525	0.583/0.373	0.515/0.291	0.696/0.481	0.591/0.336	0.532/0.263	0.739/0.448	0.639/0.298	0.587/0.229
JNET	**0.794/0.664**	**0.697/0.534**	**0.643/0.453**	**0.812/0.649**	**0.724/0.511**	**0.663/0.425**	**0.842/0.626**	**0.763/0.483**	**0.713/0.399**

network structure. In Yelp dataset, network-only solutions, i.e., DW, and text-only solutions, i.e., LDA and HFT, cannot take the full advantage of both modalities of user-generated data to capture user intents, while RTM achieved descent performance due to the integration of content and network modeling. Since the way of constructing network in StackOverflow is more content oriented, the performance of link prediction on StackOverflow prefers the text based solutions, which explains the comparable performance of LDA. Though TADW utilizes both modalities for user modeling, it fails to capture the dependency between them, leading to the poor performance on this task.

In practice, link prediction for unseen users is especially useful. For example, friend recommendation for new users in a system: they have very few or no friends, while they may associate with rich text content. This is also known as "cold-start" link prediction. Network-only solutions will suffer from the lack of information in such users. However, a joint model can overcome this limitation by utilizing user-generated text content to learn representative user vectors, thus to provide helpful link prediction results.

In order to study the models' predictive power in the cold-start setting, we randomly sampled three sets of users, regarding to the number of documents each user has, and name them as light, medium and heavy users accordingly. Each set of users consists of 200 users, and we selected 10 and 50 as the threshold for Yelp, 15 and 50 as the threshold for StackOverflow respectively. For example, in StackOverflow, light users have fewer than 15 posts, medium users have more than 15 but fewer than 50 posts, and heavy users have more than 50 posts. We compared JNET against four baselines, i.e., LDA, HFT, RTM and TADW for evaluation purpose. Because DW cannot learn representations for users without any network information, it is excluded in this experiment. We also randomly injected irrelevant users as introduced before for evaluation and we varied the ratio to change the difficulty of the task. We reported the NDCG and MAP performance on the three sets of users in Stackoverflow dataset with three different ratios, i.e., 2, 4 and 6, in Table 1, respectively.

JNET achieved consistently favorable performance in cold-start users, as accurate proximity between users is properly identified with its user representations learnt from text data. Comparing across user groups, better performance is achieved for users with more text documents. Similar results were obtained on Yelp dataset as well, but omitted due to space limit.

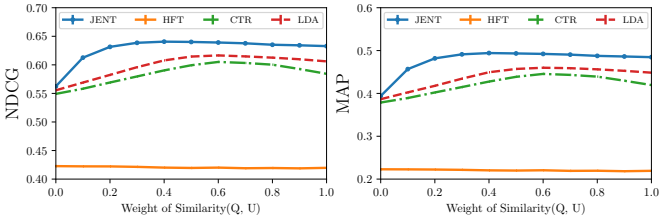


Fig. 6: Expert recommendation on StackOverflow.

1.3.4 Expert Recommendation In the sampled StackOverflow dataset, the average number of answers for questions is as low as 1.14, which indicates the difficulty for getting an expert to answer the question. If the system can suggest the right user to answer the posted questions, e.g., push the question to the selected user, more questions would be answered more quickly and accurately. We conjecture the learnt topic distribution of each question in StackOverflow, together with the identified user representation, can facilitate the task of expert recommendation for question answering. The task can be further decomposed into two components: whether the question falls into a user's skill set; and whether the user who asked the question shares similar interests with the potential candidate experts. With the learnt topic embeddings Φ and each user's embedding u_i, each user's interest over topics can be characterized as a mapping from the topic embeddings to the user's embedding, i.e., $\Phi \cdot u_i$. Together with the learnt topic distribution of each question, we can estimate the proximity between a question and a user's expertise to score the alignment between them. In the meanwhile, the closeness between users can be simply measured by the distance of their corresponding embedding vectors. As a result, the task can be formalized as finding the user that achieves the highest relatedness with the given question, where we define the relatedness as follows:

$$\text{score} = \alpha \cdot \text{cosine}(u_i \cdot \Phi, \theta_{id}) + (1 - \alpha) \cdot \text{cosine}(u_i, u_j) \tag{5}$$

Due to the limited number of answers for each question in our dataset, we selected 1,816 questions with more than 2 answers for the experiment. Besides the users that answered the given question, we also incorporated irrelevant users for each question for evaluation purpose. And the number of irrelevant users is 10 times of the number of answers. We compared against the learnt topic distributions of questions and user representations from LDA, HFT and CTR as we cannot get the topic distribution of each question from the other baselines. As we tune α, we plot the corresponding NDCG and MAP in Figure 6.

JNET achieved very promising performance in this recommendation task, as it explicitly models a user's expertise and a given question in the topic space. The estimated similarities between user-user and user-content accurately align the question to the right user. The baseline models can only capture the similarity between questions and users based on their topical similarity, which is insufficient in this task. Interestingly, as we gradually increased the weight of question-content similarity from 0 to 1, JNET's performance peaked, which indicates the relative importance between user-user and user-content similarities for this specific problem.

2 Unsupervised Machine Learning with Implicit Structure

Recent graph embedding methods achieved phenomenal success by exploiting the *neighborhood structure*. Specifically, each node is embedded by aggregating *local features* from its neighboring nodes [81];

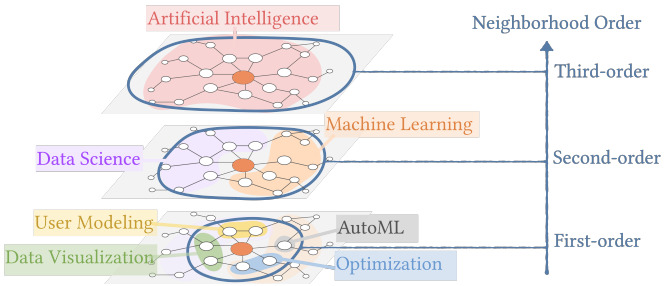


Fig. 7: An illustration of hierarchical grouping structure where higher layer identifies more global properties shared by larger groups of nodes.

then the node embedding is decoded to recover the node's *structural context*, which is composed of neighbors that can be related to this node (e.g., reachable via random walks in the original graph [134]). To effectively encode the local features and preserve the surrounding context, it is worth noting that *implicit hierarchical grouping of nodes* commonly exists in a graph's neighborhood [107, 35, 34, 108]. Nodes tend to form groups; and the groups are organized in a hierarchical manner in terms of the *neighborhood orders*: at the lower level, groups formed among low-order neighboring nodes identify fine-grained properties shared by these nodes; as higher-order neighbors are included, small groups at lower-levels are merged with globally shared patterns extracted. The same insight has been investigated in social psychology [80, 73] which suggests that related individuals (nodes) tend to flexibly manifests their properties in different groups, and new properties emerge when groups are merged into larger ones.

Figure 7 shows a motivating example on a citation graph. The nodes denoting papers are connected by citation links. The three layers concern different neighborhood orders (indicated by the blue circle) of the red target node. From the bottom up, as we include higher-order neighbors, small groups are merged to larger groups. For instance, if we inspect the first-order neighbors of the target node, we may distinguish them relating to "Optimization" or "AutoML", because more subtle distinctions between them can be identified from their node features and direct citation links. But when we consider the higher-order neighborhood, we may view nodes previously from different fine-grained groups to be members of the same but more coarse-grained groups, such as "Machine Learning" vs., "Data Science".

Although the hierarchical grouping of graph nodes is ubiquitous and informative, little attention has been devoted to it. Previous works showed that modeling a node's membership to multiple groups is effective in refining the relatedness among nodes [104, 106, 42, 130]. For example, the connection among nodes was profiled as a distribution of memberships, which is inferred from textual content [104] or extracted from random walks [106] based on each node's neighborhood [42, 130]. However, these works simply modeled the grouping of nodes with a global mixture model, ignoring the hierarchy among groups. Other attempts trying to preserve hierarchical community structure were purely based on graph structure [41, 107], and ignored its inter-dependency with node features in the neighborhood structure.

Modeling nodes' membership to hierarchical groups establishes a new principle for graph embedding: *nodes that are members of the same group should be embedded closely, and the learned node embeddings should reflect the nodes' membership to groups at different layers in the hierarchy.* In light of this principle, we define two important properties of group hierarchy in a graph: 1) the layer of the hierarchy controls the resolution of node groups, such that the higher-level layers capture global patterns shared by a broader neighborhood scope; 2) an inclusive relation exists across layers, such that lower layer groups are merged into higher layer groups carrying more shared patterns from the bottom up in the hierarchy.

To realize such a group hierarchy when exploiting neighborhood structure, we propose a novel Graph embedding model with Hierarchical Attentive Memberships, abbreviated as GraphHAM. In this model, we embed nodes and latent node groups to the same latent space, such that the affiliation of nodes toward groups can be inferred based on its context. The critical design component is an aggregation function that attends neighboring nodes guided by both node and group embeddings jointly, such that both local and global context within the neighborhood scope are captured. The node states generated for each aggregation layer are trained to recover context nodes at a certain granularity for the neighborhood scope of this layer. We further incorporate a set of structural constraints on the inferred group memberships across layers of the hierarchy, such that the inclusive relation across layers is explicitly impose for a well-defined group hierarchy.

2.1 Related Work

Hierarchical graph embedding. Recent years have witnessed numerous advances in deep architectures for learning graph embeddings, among which Graph Convolutional Networks (GCNs) received the most attention [18, 39, 81, 54, 69]. While most GCN models consider nodes as homogeneous, there are some efforts exploring the hierarchical grouping property of nodes in graphs. Hierarchical structure could be *observed* in heterogeneous graphs [110], multi-source aligned networks [68] and interconnected graphs [94]. However, in most cases the hierarchical groups are *latent*, and there are mainly two lines of studies addressing this challenge: 1) graph coarsening methods, and 2) spherical projection methods.

Graph coarsening strategies are proposed to obtain a series of successively simplified graphs capturing global patterns under different granularity [26, 197, 62, 10, 93, 203]. Graphs are coarsened by a node and edge collapsing heuristic in [26]; however, it is performed as a pre-processing step thus is isolated from later model training. Graph pooling operation is proposed to learn an intermediate weight matrix as a soft group assignment, such that nodes are merged as hyper-nodes to coarsen graphs [197, 62, 10]. However, such graph pooling methods lack necessary control on the weight matrix to maintain reasonable group hierarchy, such as the inclusive relation. *Spherical projection* methods embed low-level groups onto a spherical surface, the center of which represents the higher-level groups [41, 107]. But they only consider node co-occurrences when constructing the hierarchy. Hierarchical structure is also studied in the hyperbolic space [127]; but the learned embeddings in such space are difficult to be converted into the Euclidean vectors, which limits their applications. Hierarchical taxonomy [108] is incorporated in network embedding via nested Chinese restaurant process; however it is not directly applicable to GCN models.

Multi-membership based graph embedding. Nodes are considered to have distinct memberships to different groups depending on its local context [104, 130, 106, 132, 90]. Long et al. [106] model node memberships to topic groups on random walks in the graph. The solution heavily relies on the random walks generated prior to the actual embedding, which results in fixed memberships that are independent from the training of embedding models. Lin and Wang [104] associate each edge with multiple group channels, and propose a channel-aware attention mechanism to aggregate neighbor features based on their memberships. But external supervision from textual content in a graph is needed to infer the memberships. There are some recent studies that dynamically assign membership in each node based on a single context node [42, 130]. The selection module is trained with node embedding via Gumbel-Softmax [130] or a graph cut loss [42], but group hierarchy is still ignored.

Though the effectiveness of multi-membership modeling has been verified in the aforementioned studies, the memberships are based on a flat set of groups with the same granularity [42, 130, 106]. Our work proposes a more general model to capture nodes' membership for a hierarchy of groups under different granularity. The node groups at each layer of the hierarchy correspond to a certain neighborhood

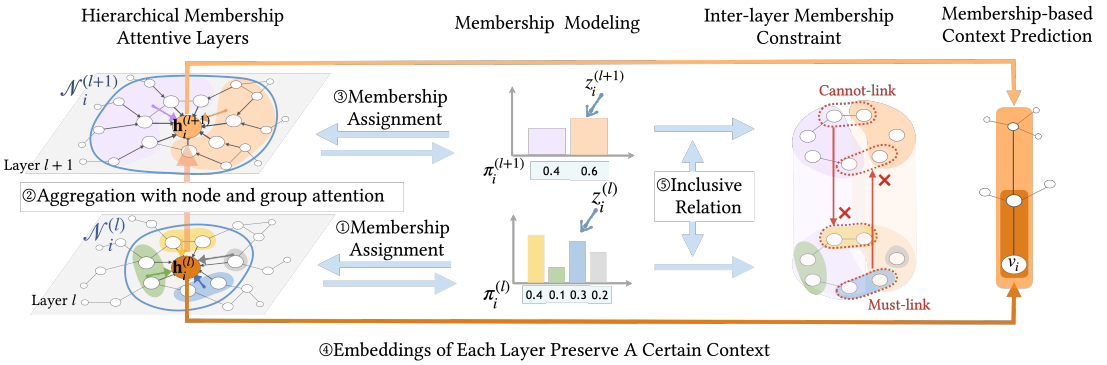


Fig. 8: Overview of GraphHAM. In each layer of the aggregation operation, a group membership is firstly sampled for each node. Then the information from neighbors is attended by the inferred membership to generate node states for the next layer. The node states for each layer are learned by recovering the context within a certain neighborhood scope. Meanwhile, inter-layer constraints are introduced to inject the inclusive relation between membership assignments across layers.

scope. It enables structured and automatic membership discovery for each node, which supports more comprehensive and accurate node embeddings.

2.2 Model: GraphHAM

In this section, we explain the detailed design of our model GraphHAM as illustrated in Figure 8. Each node is modeled as a mixture of groups at each layer (Section 2.2.1). The hierarchical membership attentive layer aggregates information from neighboring nodes with both node-level and group-level attentions (Section 2.2.2). Finally, the node embedding vectors on each layer are learned by preserving the structural contexts within a certain neighborhood scope (Section 2.2.3). We further impose structural constraints to regularize the inferred memberships across layers (Section 2.2.4).

2.2.1 Membership Modeling We propose to infer nodes' latent affiliation to each group based on their embeddings. Then each time a group assignment is sampled for a node from its membership when it interacts with another node, adopting the intuition that the group membership of each node is *context dependent* and its particular assignment is only manifested under a specific context.

Given graph $G = (V, E)$ with V denoting a set of nodes and E representing the edges, we assume that each node v_i is associated with an embedding vector $\mathbf{h}_i \in \mathbb{R}^d$ that depicts the states of this node. As illustrated in Figure 7, groups can be formed by nodes that share similar states, when we view node proximity in a certain neighborhood scope. We assume that each node is a mixture of K groups with a corresponding group-membership distribution denoted by $\pi_i \in \Delta^K$, where Δ^K is a probability simplex over K dimensions, i.e., $\forall k, \pi_i^k \geq 0$ and $\sum_{k=1}^K \pi_i^k = 1$. To infer the latent membership of each node, we embed each group in the same space as node state denoted by $\phi_k \in \mathbb{R}^d$, and use $\Phi \in \mathbb{R}^{K \times d}$ to represent the matrix of embeddings of all K groups. Therefore, given a node state $\mathbf{h}_i$, we measure the node's affiliation to each group via $\Phi \cdot \mathbf{h}_i \in \mathbb{R}^{K \times 1}$. We then sample a group assignment for each node denoted as z_i, following the commonly adopted assumption that each node only manifests a single membership depending on the specific context [5, 130]. We summarize the procedure in the following steps.

- For each node v_i:
  - Draw its membership vector $\pi_i \sim \text{Dir}(\text{softmax}(\Phi \cdot \mathbf{h}_i))$
  - Draw its group assignment $z_i \sim \text{Multi}(\pi_i)$

where $\text{Dir}(\cdot)$ denotes the Dirichlet distribution and $\text{Multi}(\cdot)$ is the Multinomial distribution. Note that the hard assignment based on a categorical distribution is non-differentiable, which blocks the flow of gradients in later end-to-end optimization. We adopt the Gumbel-Softmax trick [97] to reparameterize the multinomial distribution and draw a one-hot assignment z_i as follows:

$$z_i = \text{one-hot}(\text{argmax}_k[\pi_{i,k} + g]), g \sim \text{Gumbel}(0, 1)$$

where g is sampled from the standard Gumbel distribution. The non-differentiable $\text{argmax}(\cdot)$ operation is further replaced with softmax to render the whole process differentiable:

$$z_i \propto \exp\big((\pi_{i,k} + g)/\tau\big) \tag{6}$$

where τ is the temperature parameter to control the extent to which the output approximates the $\text{argmax}(\cdot)$ operation: As $\tau \to 0$, samples from the Gumbel-Softmax distribution become one-hot.

Multiple sets of groups can be modeled when considering nodes with different scopes of neighborhood. As shown in Figure 8, for layer $l+1$ concerning neighborhood scope $\mathcal{N}^{(l+1)}$, the target node marked by red is associated with a group-membership distribution $\pi_i^{(l+1)}$ obtained by group embeddings $\Phi^{(l+1)} \in \mathbb{R}^{K^{(l+1)} \times d^{(l+1)}}$ and node states $\mathbf{h}_i^{(l+1)} \in \mathbb{R}^{d^{(l+1)}}$. It is then assigned a new group $z_i^{(l+1)}$. Since the higher-level layer captures more coarse-grained groups covering larger neighborhood scope, the number of groups should become smaller, i.e., $K^{(l)} > K^{(l+1)}$.

2.2.2 Hierarchical Membership Attentive Layers Our principle suggests that node proximity in the embedding space should reflect their closeness in the group hierarchy. Following this insight, we propose an aggregation function that attends neighboring nodes by both node-level and group-level relatedness.

Formally, the aggregation operation of graph convolutional layer $l+1$ takes two inputs: 1) **node states** from the previous layer l, $\{\mathbf{h}_1^{(l)}, \ldots, \mathbf{h}_{|V|}^{(l)}\}$, where $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$; 2) **group states** within the neighborhood scope $\mathcal{N}^{(l)}$, which is profiled by the matrix of group embeddings $\Phi^{(l)}$. The layer $l+1$ aggregates information from neighborhood scope $\mathcal{N}^{(l)}$ and generates a new set of node states $\{\mathbf{h}_1^{(l+1)}, \ldots, \mathbf{h}_{|V|}^{(l+1)}\}, \mathbf{h}_i' \in \mathbb{R}^{d^{(l+1)}}$. We stack multiple layers to capture the information from neighborhoods of different scopes, where the node states output by a lower layer are used as input to the layer above it. We denote the raw input node features as $\mathbf{h}^{(1)}$, the first-order neighbor scope as $\mathcal{N}^{(1)}$, and stack L layers.

When encoding the target node v_i, we propose the following aggregation function to emphasize neighboring nodes that have similar states and belong to related groups:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\frac{1}{M}\sum\nolimits_{m=1}^{M}\sum\nolimits_{j \in \mathcal{N}_i} \lambda_{ij}^m \alpha_{ij}^m \mathbf{W}^{(l+1),m} \mathbf{h}_j^{(l)}\right) \tag{7}$$

where $\mathcal{N}_i$ is the immediate neighborhood of node i, $\{\mathbf{W}^{(l+1),m} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}\}_{m=1}^M$ is a set of state transformation matrices and σ is a non-linear function. The states of neighboring nodes are re-weighed by both a group-level coefficient λ_{ij}^m and a node-level coefficient α_{ij}^m, which are calculated by multi-head attention [174]. Specifically, we calculate the following attention weights with a shared transformation

parameterized by $\mathbf{W}^{(l+1),m}$ for each head m:

$$\lambda_{ij}^m = \text{att}\left(\mathbf{W}^{(l+1),m}\Phi^{(l)}_{z_i^{(l)}}, \mathbf{W}^{(l+1),m}\Phi^{(l)}_{z_j^{(l)}}\right), \alpha_{ij}^m = \text{att}\left(\mathbf{W}^{(l+1),m}\mathbf{h}_i^{(l)}, \mathbf{W}^{(l+1),m}\mathbf{h}_j^{(l)}\right)$$

Thus λ_{ij}^m captures the global relatedness between the two nodes' assigned groups, while α_{ij}^m measures the local relatedness in terms of node states. The attention function $\text{att}(\cdot,\cdot)$ [6] can be expressed as follows by normalizing over all nodes within the neighborhood:

$$\text{att}(\mathbf{p}_i, \mathbf{p}_j) = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{p}_i \| \mathbf{p}_j])\right)}{\sum_{j' \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\big(\mathbf{a}^\top[\mathbf{p}_i \| \mathbf{p}'_j]\big)\right)}$$

where $\|$ is the concatenation operation over two vectors, and $\mathbf{a} \in \mathbb{R}^{2d^{(l)}}$ is the weight vector of a linear transformation.

The combination of group-level and node-level attention is the key to realizing our principle: if two nodes have similar node states measured by α, and belong to related groups indicated by λ, more information should be passed between them; and thus they are encoded closer in the embedding space. As a result, the relatedness of nodes with respect to the hierarchy of groups is preserved in the embedding space.

2.2.3 Membership-based Context Prediction Our proposed principle argues that node embeddings should preserve the group hierarchy, which requires us to capture node proximity at each layer of the hierarchy. This can be achieved by aligning the neighborhood scope when aggregating information and decoding the context. The former is achieved by our membership-aware attentive layer introduced above, and now we introduce our membership-based context decoder to preserving graph hierarchy.

Given a target node v_i, the context summarizes its surrounding nodes when it manifests a certain membership. We introduce a trainable membership-based context vector $\mathbf{Q}^{(l)}_{j,z_i^{(l)}} \in \mathbb{R}^{d^{(l)}}$ to encode each node v_j that constitutes the context of the target node v_i within the neighborhood scope $\mathcal{N}_i^{(l)}$. This context vector can be decoded by maximizing the likelihood of observing the context nodes given a target node, defined by the following skip-gram based objective [130]:

$$\begin{aligned}\mathcal{L}^{(l)}_{context} &= \sum_{v_i \in V} \sum_{j \in \mathcal{N}_i^{(l)}} -\log p\big(v_j | v_i, z_i^{(l)}\big) \\ &= \sum_{v_i \in V} \sum_{j \in \mathcal{N}_i^{(l)}} -\log\left(\sigma(\mathbf{h}_i^{(l)\top} \cdot \mathbf{Q}^{(l)}_{j,z_i^{(l)}})\right) - \mathbb{E}_{j_n \sim \bar{\mathcal{N}}_i^{(l)}} \log\left(\sigma(-\mathbf{h}_i^{(l)\top} \cdot \mathbf{Q}^{(l)}_{j_n,z_i^{(l)}})\right)\end{aligned} \tag{8}$$

The context vector $\mathbf{Q}^{(l)}_{j,z_i^{(l)}}$ depends on the group assignment $z_i^{(l)}$ of the target node v_i, which aligns with our intuition that a single membership is manifested in a given context. $\bar{\mathcal{N}}_i^{(l)}$ denotes the set of node outside the immediate neighborhood of v_i. This objective represents the context reconstruction error, such that nodes that frequently co-occur within scope $\mathcal{N}^{(l)}$ should be pushed closer in the embedding space. We use negative sampling to construct $\bar{\mathcal{N}}_i^{(l)}$ for efficient calculation. The membership-based context vector and the model parameters are jointly learned.

As illustrated in Figure 8, node state generated by each layer is decoded to recover the context composed of the neighborhood with corresponding scope (indicated by orange arrows). By aligning the neighborhood scope in the aggregation layer and the decoded context, we capture the anticipated property of graph hierarchy that each layer controls the resolution of groups.

2.2.4 Inter-layer Membership Constraints An inclusive relation exists in the hierarchy of groups to depict the emergence of groups bottom up. We leverage this relation to explicitly introduce inter-layer constraints to regularize the modeling of latent group memberships of nodes.

We define two sets of constraints: 1) **must-link** denoted by $\mathcal{M}^{(l+1)} = \{(v_i, v_j) : z_i^{(l)} = z_j^{(l)}\}$, which implies that nodes v_i and v_j should be members of the the same group in the higher layer $l+1$, as they already belong to the same group in the lower layer; 2) **cannot-link** denoted by $\mathcal{C}^{(l)} = \{(v_i, v_j) : z_i^{(l+1)} \neq z_j^{(l+1)}\}$, which suggests that these two nodes should belong to different groups in the lower layer l, since they belong to different groups in the higher layer. We then introduce the following regularization term to penalize the violation of must-link and cannot-link constraints:

$$\mathcal{L}_{reg}^{(l)} = \sum_{(v_i,v_j)\in\mathcal{M}^{(l+1)}} \gamma \cdot \mathbb{1}\left(z_i^{(l+1)} \neq z_j^{(l+1)}\right) + \sum_{(v_i,v_j)\in\mathcal{C}^{(l)}} \beta \cdot \mathbb{1}\left(z_i^{(l)} = z_j^{(l)}\right) \tag{9}$$

where $\mathbb{1}(\cdot)$ is the indicator function. The strength of penalty is controlled by γ and β respectively.

The node pairs marked by red circles in Figure 8 illustrate the purpose of the inter-layer membership constraints, where two pairs of nodes are penalized for violating the must-link and cannot-link requirements on membership assignments. This regularization ensures the dependency between memberships across layers, such that node proximity maintains the consistency in the hierarchy.

Applying the inter-layer constraints to the loss of recovering membership-based contexts with different neighborhood scopes, we obtain the final optimization objective as follows to learn the node states generated on each aggregation layer in GraphHAM:

$$\mathcal{L} = \sum\nolimits_{l=1}^{L} \mathcal{L}_{context}^{(l)} + \sum\nolimits_{l=1}^{L-1} \mathcal{L}_{reg}^{(l)} \tag{10}$$

To prepare the embeddings for use on downstream tasks, such as node classification, we concatenate the layer-wise node states to obtain a final embedding of each node, which can be further fine-tuned by introducing a task-specific loss to Eq (10).

2.3 Experiments

We evaluated GraphHAM on two popular tasks, node classification and link prediction, to verify its effectiveness in preserving node property and graph structure (Section 2.3.1 and 2.3.2). In the qualitative analysis, we mapped the learned node embeddings to a 2-D space to demonstrate the membership inferred by GraphHAM, which verified the effectiveness of the learned group-membership distributions in discovering nodes at the boundary of groups (Section 2.3.3). Finally, we analyzed GraphHAM via a comprehensive ablation study which verified the effectiveness of membership attentive layers and inter-layer membership regularization (Section 2.3.4).

• **Datasets.** We included six public datasets for our evaluation, ranging from academic citation networks to large-scale social networks. The citation network datasets, Cora, Citeseer and Pubmed [152], contain research papers as nodes and citation links as edges. The Facebook dataset [148] represents official Facebook homepages as nodes and mutual likes between them as edges. The Youtube dataset [168] includes users as nodes and co-subscription relations as edges. The Amazon graph [29] denotes products as nodes which are connected by edges if purchased together. Table 2 shows detailed statistics of the datasets, and the clustering coefficient measures the degree to which nodes tend to be clustered together.

• **Baselines.** The proposed GraphHAM is compared against a rich collection of graph embedding models: 1) **GraphSage** [54] uniformly passes information through edges without neighborhood-based attention.

Table 2: Statistics of evaluation graph benchmark datasets.

Dataset	#Node	#Edge	#Class	Clustering Coef.
Cora	2,708	5,429	7	0.241
Citeseer	3,327	4,732	6	0.141
Pubmed	19,717	44,338	3	0.060
Facebook	22,470	171,002	4	0.360
Youtube	1,138,499	2,990,443	47	0.080
Amazon	2,449,029	123,718,280	47	0.419

2) **GAT** [174] incorporates attention mechanism on node states to reweigh neighboring nodes when aggregating their information. 3) **GraphSTONE** [106] is a multi-membership baseline, but ignores the hierarchy of groups. It extracts structural patterns as groups from random walks, and uses a topic model to infer group membership for reweighing neighbor nodes. 4) **DeepMinCut** [42] is a multi-membership baseline which derives nodes' membership assignments by minimizing a graph cut loss. 5) **asp2vec** [130] is a multi-membership baseline which dynamically assigns each node a membership based on its context in random walk. 6) **H-GCN** [62] is a hierarchical embedding baseline, which coarsens graphs by learning a soft membership assignment matrix for each aggregation layer, and then produces node-level embeddings by refining the layers under node classification loss. 7) **GNE [41]** is a hierarchical embedding baseline based on spherical projection, which projects lower-level groups to a sphere with the center representing the merged higher-level groups. 8) **SpaceNE [107]** projects groups into different subspaces whose dimensionalities reflect the hierarchy, such that groups in lower-dimension subspace can be merged in higher-dimension subspace.

• **Experiment settings.** We set the node embedding size $d = 128$, and use $L = 2$ aggregation layers for all GCN-based methods. Each layer in GraphHAM has its dimension set to $d_1 = d_2 = d/L = 64$. The models are trained in a mini-batch manner following [54]. For each node in a batch, we sample $S_1 = S_2 = 25$ neighbors for each layer. To sample node pairs for skip-gram optimization, we conduct random walks from each node 50 times with a window size set to 2. We set the number of negative samples equal to the number of positive examples. The other parameters of baselines are set to their optimal values as suggested in their original papers. All the results are reported based on five-fold cross-validation.

• **Model complexity.** Compared with commonly used GCNs, we only introduced two sets of additional parameters to model group membership: the membership embeddings $\Phi^{(l)} \in \mathbb{R}^{K^{(l)} \times d^{(l-1)}}$ for each aggregation layer, and the context node embeddings $\mathbf{Q}^{(l)} \in \mathbb{R}^{K^{(l)} \times d^{(l)}}$ for the membership-based context decoder. In general, GCNs are equipped with a weight matrix of state transformation $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ for each layer. Since the number of groups K is usually set smaller than the embedding dimension d, we did not significantly increase model complexity in GraphHAM.

2.3.1 Node Classification The node embeddings are fed to predict the node labels. Recall that each aggregation layer of GraphHAM produces a vector for each node to encode neighbor information within a given scope. We concatenate these vectors from all layers to serve the node classification task. The composed embedding is trained in a multitask manner by joining the classification loss with Eq (10).

Table 3 summarizes the performance of GraphHAM against baseline models with *accuracy, micro- and macro- F1 score* metrics. We can clearly observe that GraphHAM consistently outperformed baselines on these datasets. Compared with GraphSage and GAT which do not model group membership, GraphHAM achieved a significant improvement. This demonstrates the importance of modeling the latent group structure in graphs. GraphSTONE discovered global structures by summarizing random walk

Table 3: Performance comparisons of node classification task under different metrics.

Dataset	Metric	GraphSage	GAT	H-GCN	GraphSTONE	GNE	SpaceNE	DeepMinCut	**GraphHAM**
Cora	Accuracy	$0.812_{\pm 0.014}$	$0.829_{\pm 0.013}$	$0.835_{\pm 0.013}$	$0.823_{\pm 0.016}$	$0.787_{\pm 0.015}$	$0.796_{\pm 0.011}$	$0.830_{\pm 0.013}$	$\mathbf{0.853}_{\pm 0.014}$
	Micro-F1	$0.823_{\pm 0.014}$	$0.834_{\pm 0.013}$	$0.844_{\pm 0.014}$	$0.831_{\pm 0.015}$	$0.782_{\pm 0.019}$	$0.788_{\pm 0.018}$	$0.840_{\pm 0.014}$	$\mathbf{0.860}_{\pm 0.013}$
	Macro-F1	$0.785_{\pm 0.013}$	$0.800_{\pm 0.013}$	$0.804_{\pm 0.013}$	$0.798_{\pm 0.015}$	$0.755_{\pm 0.011}$	$0.746_{\pm 0.019}$	$0.806_{\pm 0.014}$	$\mathbf{0.824}_{\pm 0.013}$
Citeseer	Accuracy	$0.703_{\pm 0.015}$	$0.723_{\pm 0.014}$	$0.724_{\pm 0.013}$	$0.720_{\pm 0.015}$	$0.676_{\pm 0.016}$	$0.679_{\pm 0.013}$	$0.723_{\pm 0.014}$	$\mathbf{0.727}_{\pm 0.014}$
	Micro-F1	$0.756_{\pm 0.014}$	$0.772_{\pm 0.014}$	$0.781_{\pm 0.013}$	$0.768_{\pm 0.016}$	$0.722_{\pm 0.011}$	$0.725_{\pm 0.010}$	$0.723_{\pm 0.015}$	$\mathbf{0.783}_{\pm 0.013}$
	Macro-F1	$0.724_{\pm 0.015}$	$0.739_{\pm 0.014}$	$\mathbf{0.748}_{\pm 0.013}$	$0.739_{\pm 0.015}$	$0.706_{\pm 0.012}$	$0.710_{\pm 0.010}$	$0.682_{\pm 0.016}$	$0.746_{\pm 0.014}$
Pubmed	Accuracy	$0.788_{\pm 0.015}$	$0.797_{\pm 0.014}$	$0.797_{\pm 0.014}$	$0.794_{\pm 0.016}$	$0.749_{\pm 0.011}$	$0.756_{\pm 0.011}$	$0.723_{\pm 0.015}$	$\mathbf{0.812}_{\pm 0.015}$
	Micro-F1	$0.802_{\pm 0.014}$	$0.813_{\pm 0.014}$	$0.817_{\pm 0.015}$	$0.809_{\pm 0.016}$	$0.763_{\pm 0.019}$	$0.775_{\pm 0.010}$	$0.723_{\pm 0.015}$	$\mathbf{0.824}_{\pm 0.015}$
	Macro-F1	$0.794_{\pm 0.014}$	$0.801_{\pm 0.015}$	$0.813_{\pm 0.015}$	$0.804_{\pm 0.017}$	$0.755_{\pm 0.013}$	$0.762_{\pm 0.011}$	$0.712_{\pm 0.011}$	$\mathbf{0.819}_{\pm 0.014}$
Facebook	Accuracy	$0.875_{\pm 0.016}$	$0.894_{\pm 0.017}$	$0.901_{\pm 0.016}$	$0.905_{\pm 0.018}$	$0.864_{\pm 0.017}$	$0.848_{\pm 0.018}$	$0.876_{\pm 0.017}$	$\mathbf{0.918}_{\pm 0.016}$
	Micro-F1	$0.894_{\pm 0.015}$	$0.907_{\pm 0.016}$	$0.915_{\pm 0.016}$	$0.918_{\pm 0.017}$	$0.867_{\pm 0.014}$	$0.852_{\pm 0.012}$	$0.879_{\pm 0.016}$	$\mathbf{0.930}_{\pm 0.016}$
	Macro-F1	$0.889_{\pm 0.015}$	$0.905_{\pm 0.015}$	$0.909_{\pm 0.016}$	$0.912_{\pm 0.017}$	$0.853_{\pm 0.015}$	$0.846_{\pm 0.014}$	$0.865_{\pm 0.016}$	$\mathbf{0.924}_{\pm 0.016}$
Youtube	Accuracy	$0.732_{\pm 0.014}$	$0.745_{\pm 0.014}$	$0.744_{\pm 0.015}$	$0.739_{\pm 0.016}$	$0.713_{\pm 0.017}$	$0.722_{\pm 0.012}$	$0.742_{\pm 0.017}$	$\mathbf{0.755}_{\pm 0.015}$
	Micro-F1	$0.775_{\pm 0.014}$	$0.787_{\pm 0.015}$	$0.782_{\pm 0.015}$	$0.784_{\pm 0.016}$	$0.753_{\pm 0.016}$	$0.759_{\pm 0.014}$	$0.767_{\pm 0.017}$	$\mathbf{0.795}_{\pm 0.015}$
	Macro-F1	$0.696_{\pm 0.014}$	$0.710_{\pm 0.015}$	$0.708_{\pm 0.015}$	$0.711_{\pm 0.015}$	$0.677_{\pm 0.014}$	$0.681_{\pm 0.011}$	$0.701_{\pm 0.016}$	$\mathbf{0.721}_{\pm 0.016}$
Amazon	Accuracy	$0.659_{\pm 0.015}$	$0.674_{\pm 0.015}$	$0.652_{\pm 0.016}$	$0.662_{\pm 0.017}$	$0.612_{\pm 0.013}$	$0.632_{\pm 0.010}$	$0.667_{\pm 0.015}$	$\mathbf{0.686}_{\pm 0.016}$
	Micro-F1	$0.727_{\pm 0.015}$	$0.743_{\pm 0.015}$	$0.731_{\pm 0.016}$	$0.733_{\pm 0.017}$	$0.687_{\pm 0.019}$	$0.703_{\pm 0.010}$	$0.705_{\pm 0.015}$	$\mathbf{0.742}_{\pm 0.015}$
	Macro-F1	$0.297_{\pm 0.014}$	$0.325_{\pm 0.015}$	$0.295_{\pm 0.015}$	$0.305_{\pm 0.016}$	$0.254_{\pm 0.015}$	$0.267_{\pm 0.011}$	$0.303_{\pm 0.018}$	$\mathbf{0.342}_{\pm 0.015}$

Table 4: Performance on link prediction task.

Dataset	Metric	GraphSage	GraphSTONE	asp2vec	**GraphHAM**
Cora	AUC	$0.849_{\pm 0.018}$	$0.858_{\pm 0.015}$	$0.865_{\pm 0.021}$	$\mathbf{0.869}_{\pm 0.016}$
	MRR	$0.674_{\pm 0.019}$	$0.680_{\pm 0.022}$	$0.683_{\pm 0.023}$	$\mathbf{0.692}_{\pm 0.018}$
Citeseer	AUC	$0.935_{\pm 0.015}$	$0.944_{\pm 0.017}$	$0.944_{\pm 0.015}$	$\mathbf{0.957}_{\pm 0.016}$
	MRR	$0.762_{\pm 0.018}$	$0.768_{\pm 0.017}$	$0.741_{\pm 0.013}$	$\mathbf{0.778}_{\pm 0.015}$
Pubmed	AUC	$0.953_{\pm 0.017}$	$\mathbf{0.962}_{\pm 0.020}$	$0.927_{\pm 0.014}$	$0.959_{\pm 0.015}$
	MRR	$0.886_{\pm 0.019}$	$0.902_{\pm 0.017}$	$0.841_{\pm 0.018}$	$\mathbf{0.906}_{\pm 0.016}$
Facebook	AUC	$0.954_{\pm 0.019}$	$0.965_{\pm 0.020}$	$\mathbf{0.969}_{\pm 0.016}$	$0.966_{\pm 0.018}$
	MRR	$0.832_{\pm 0.016}$	$0.844_{\pm 0.022}$	$0.842_{\pm 0.024}$	$\mathbf{0.855}_{\pm 0.020}$
Youtube	AUC	$0.757_{\pm 0.014}$	$0.763_{\pm 0.016}$	$0.750_{\pm 0.013}$	$\mathbf{0.768}_{\pm 0.014}$
	MRR	$0.594_{\pm 0.022}$	$0.601_{\pm 0.18}$	$0.587_{\pm 0.020}$	$\mathbf{0.608}_{\pm 0.019}$
Amazon	AUC	$0.792_{\pm 0.015}$	$0.808_{\pm 0.013}$	$0.818_{\pm 0.016}$	$\mathbf{0.824}_{\pm 0.014}$
	MRR	$0.579_{\pm 0.017}$	$0.584_{\pm 0.015}$	$0.589_{\pm 0.022}$	$\mathbf{0.593}_{\pm 0.015}$

patterns; but GraphHAM still outperformed it, which proves the effectiveness of inferring group membership in an end-to-end fashion. DeepMinCut modeled global structures of the graph via graph cut, but it was less effective than our method because node features and the grouping hierarchy were ignored in DeepMinCut. H-GCN proposed successive pooling operations which captured the grouping hierarchy, but lacked the control on the group assignments to calibrate the structure across layers. GNE and SpaceNE recursively merged lower-level groups into higher-level groups in an unsupervised manner and ignored node features, thus gave worse performance.

2.3.2 Link Prediction The task is to predict the linkage between two nodes via the similarity of their embeddings [156, 195, 204]. We formulate the task as a ranking problem to retrieve linked nodes from a candidate set with negative (irrelevant) nodes. We utilize *Area under the ROC Curve (AUC)* and *scaled mean reciprocal rank (MRR)* metrics to demonstrate how effectively the models can rank real neighbors at higher positions. Since the links between nodes correspond to the context defined by the first-order neighborhood, we use the embedding vectors generated by the first layer in GraphHAM to make the prediction.

Table 4 summarizes the results of the link prediction task. GraphSAGE used two hidden layers to aggregate second-order neighborhood, which is empirically better than using one layer in it. In contrast, despite using only the output of a single hidden layer, GraphHAM still outperformed GraphSAGE, which

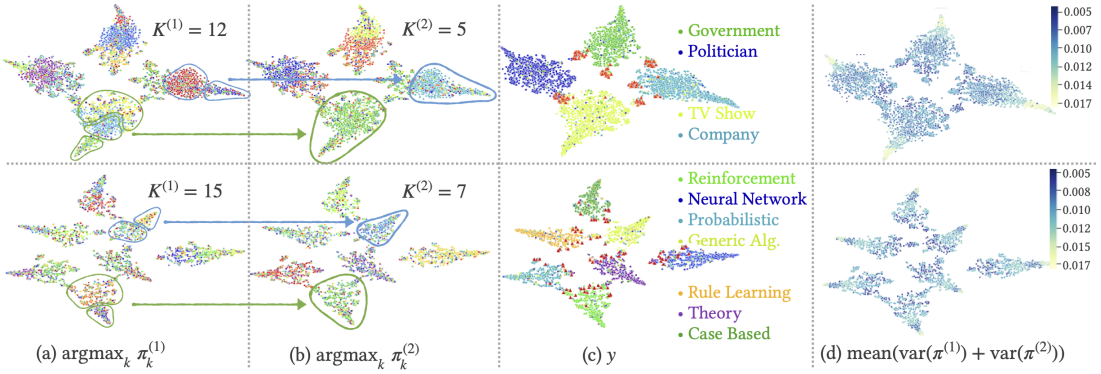


Fig. 9: Visualization of the learned embeddings on Facebook (upper) and Cora (lower) graph. Red triangles denote the boundary nodes with small variance of π. The rest colors denote the ground-truth classes (Y) or sampled group assignment across two layers ($z^{(1)}, z^{(2)}$) on each node.

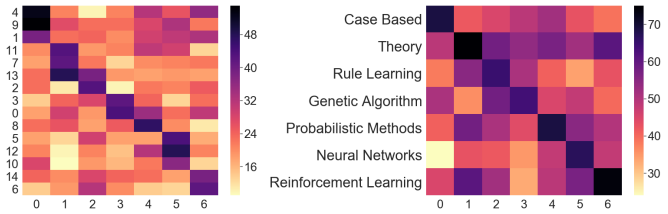


Fig. 10: Heatmaps about the degree of correlation between group assignments $z^{(1)}, z^{(2)}$ (left) and between $z^{(2)}, y$ (right) on Cora, measured by their concurrency on nodes.

strongly suggests the effectiveness of our group membership modeling in capturing global patterns. GraphSTONE and asp2vec modeled multiple memberships, but ignored the inherent hierarchy concerning different granularity of contexts, and thus were still outperformed by GraphHAM.

2.3.3 Proof-of-Concept Visualization To analyze the quality of the jointly learned node embeddings and groups from GraphHAM, we use the t-SNE algorithm to project the composed node embeddings to a 2-D space, and visualize Facebook graph in the first row and Cora graph in the second row in Figure 9. The node color has different meanings across the columns. In column (a) and (b), the color denotes the group assignment of each node v_i with the largest affiliation strength in the first and second layer, i.e., $\text{argmax}_k \pi_{i,k}^{(1)}$ and $\text{argmax}_k \pi_{i,k}^{(2)}$, respectively. In column (c), the color shows the ground-truth label with its definition listed aside. In column (d), we calculate the average variance on $\pi^{(1)}$ and $\pi^{(2)}$ for each node to measure its degree of concentration over groups, and the color reflects the conentration: a darker color means a lower degree of concentration, which suggests that the node's affiliation to different groups is evenly distributed. Meanwhile, the nodes with the lowest concentration degree are highlighted by red triangles in column (c) for the purpose of illustration.

To better demonstrate the correlation between the learned groups and inferred memberships across layers and the ground-truth labels, we also visualize their co-occurrence matrices on Cora dataset, shown in Figure 10. In the left heatmap, for each entry indexed by row i and column j, we calculate the number of nodes that are concurrently assigned with $z^{(1)} = i$ in the first layer and $z^{(2)} = j$ in the second layer. In the right heatmap, we count the number of nodes that have ground-truth label $y = i$ and are assigned

Table 5: Ablation study of three model variants on node classification task. The results are performance gap between each variant and the complete GraphHAM. * suggests p-value < 0.05.

Dataset	Metric	GraphHAM-λ	GraphHAM-$\mathbf{Q}$	GraphHAM-$\mathcal{L}_{reg}$
Cora	Accuracy	$\mathbf{-0.017}^*$	-0.007^*	-0.012^*
	Micro-F1	$\mathbf{-0.021}^*$	-0.016^*	-0.020^*
	Macro-F1	$\mathbf{-0.012}^*$	-0.007^*	-0.008^*
Citeseer	Accuracy	$\mathbf{-0.004}$	-0.003	$\mathbf{-0.004}$
	Micro-F1	-0.008^*	-0.003	$\mathbf{-0.009}^*$
	Macro-F1	-0.005	-0.002	$\mathbf{-0.007}^*$
Pubmed	Accuracy	$\mathbf{-0.017}^*$	-0.013^*	-0.007^*
	Micro-F1	$\mathbf{-0.012}^*$	-0.009^*	-0.003
	Macro-F1	$\mathbf{-0.012}^*$	-0.008^*	-0.004
Facebook	Accuracy	$\mathbf{-0.020}^*$	-0.010^*	-0.013^*
	Micro-F1	$\mathbf{-0.016}^*$	-0.011^*	-0.012^*
	Macro-F1	-0.012^*	-0.009^*	$\mathbf{-0.015}^*$
Youtube	Accuracy	$\mathbf{-0.007}^*$	-0.006	-0.003
	Micro-F1	$\mathbf{-0.007}^*$	-0.005	-0.002
	Macro-F1	$\mathbf{-0.009}^*$	-0.006	-0.003
Amazon	Accuracy	-0.008^*	-0.003	$\mathbf{-0.012}^*$
	Micro-F1	-0.010^*	-0.002	$\mathbf{-0.013}^*$
	Macro-F1	-0.007^*	-0.003	$\mathbf{-0.009}^*$

to group $z^{(2)} = j$ in the second layer. The color denotes the number of nodes satisfying those respective assignments, thereby reflecting the degree of correlation between $z^{(1)}, z^{(2)}$ and y.

The visualizations in Figure 9 and Figure 10 together demonstrate two intriguing properties of GraphHAM as discussed below.

• **A hierarchy of groups is captured.** Comparing column (a) and (b) of Figure 9, we can clearly observe different node groups are captured. And more interestingly, a hierarchy of groups is automatically discovered: in the highlighted circles, different groups of nodes shown in column (a) are merged to form larger groups shown in column (b). The trend of merging groups is a clear manifestation of the desired group hierarchy, where coarse-grained properties shared by a larger group of nodes are captured when we aggregate information from lower-level groups with fine-grained properties. Comparing column (b) and (c), we show that the learned node membership is also well aligned with node labels, which suggests that node label as a comprehensive signal to distinguish nodes is captured by the hierarchical group modeling. The correlation among $z^{(1)}, z^{(2)}$ and y reported in Figure 10 also supports our argument. From the diagonals with large number of nodes, we observe a clear merging structure, where multiple entries of $z^{(1)}$ frequently co-occur with one specific entry in $z^{(2)}$. This suggests a trend of group merging from lower to higher layers.

• **Our membership modeling discovers nodes at the boundary of groups.** Recall that we calculate the variance of π to measure the concentration of nodes' group affiliation. A low variance means that π is flat such that the node's affiliation to different groups is evenly distributed. In other words, such nodes have no strong ties to any group and therefore reside at the boundary of circles [1]. In column (c) of Figure 9, we observe that the nodes marked by red triangles with the lowest variance are indeed located at the edge of different groups. The color gradient in Column (d) clearly demonstrates the coherence between the concentration degree over π and the position of nodes in groups: the concentration decreases as we view from the center of a group to its boundary. This shows that modeling group membership encodes global structure, thereby endowing the learned embeddings with collective patterns in addition to the local pairwise proximity between nodes.

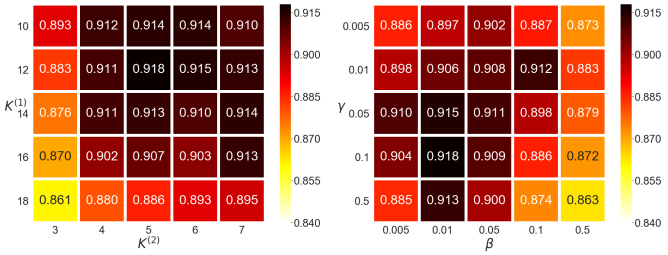


Fig. 11: Accuracy of node classification on Facebook graph under different hyper-parameter settings for group size $K^{(1)}, K^{(2)}$ (left) and regularization coefficients γ, β.

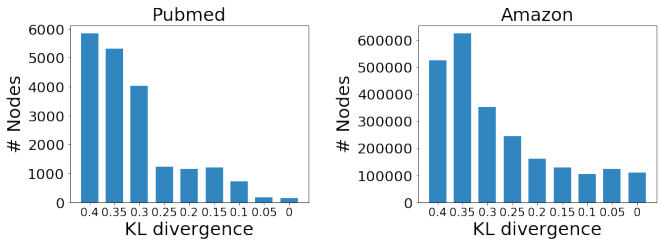


Fig. 12: Number of nodes fall into each interval of KL divergence between self attention coefficients λ and group attention coefficients α.

2.3.4 Model Analysis We provide an overall analysis on GraphHAM, including an ablation study verifying the effectiveness of each component of it and a sensitivity study about the hyper-parameters related to membership modeling.

• **Ablation Study.** We construct three variants of GraphHAM by disabling one component at a time: 1) **GraphHAM-λ** removes the membership-based attention coefficient λ in the aggregation layer defined in Eq Eq (7); 2) **GraphHAM-Q** omits the group indicator z_i in Eq (8) and replaces $\mathbf{Q}$ with a single vector $\mathbf{q}$, thus only preserves a membership-agnostic context; 3) **GraphHAM-$\mathcal{L}_{reg}$** removes the inter-layer membership regularization defined by Eq (9), and thus no inclusive constraint is imposed on groups across layers in it. The performance of these variants compared with the complete model on node classification is summarized in Table 5. We use Student's t-test to quantify the difference between the cross-validation results from the complete GraphHAM model and each variant. The values marked with asteroid in the table suggest the difference is significant (i.e., p-value¡0.05). We can verify the importance of each component based on the gap in performance. First, GraphHAM-λ gave the worst results and the majority of performance values were significantly different from the complete model (marked by star), which highlights the importance of the membership-level attention in encoding the global information. The second most effective design is the inter-layer regularization that explicitly forces an inclusive relation across layers to form a well-defined hierarchy. The membership-dependent context decoder also improves the embedding quality, which suggests that even the same context perceived by nodes with different memberships reveal different information about node neighborhood.

• **Group Attention versus Node Attention.** Since the membership-based attention gives the most improvement, an interesting question to study is *how different the group-level and the node-level attentions are*. To quantify the information difference brought by these two types of attention, on each node v_i, we calculated the average KL-divergence between α and λ as follows:

$$\text{diff}(v_i) = \frac{1}{|\mathcal{N}_i|} \sum\nolimits_{j \in \mathcal{N}_i} KL(\alpha_{ij}, \lambda_{ij})$$

We then count the number of nodes in different ranges of KL-divergence shown in Figure 12, where each bar x collects nodes with $\text{diff}(v) \in [x, x + 0.05)$. It is clear that a large amount of nodes exhibit high degree of difference, which demonstrates that these two types of attention capture different aspects of node relatedness and are thus complementary to each other.

• **Hyper-Parameter Sensitivity.** We analyzed two groups of hyper-parameters related to membership modeling: the numbers of groups $K^{(1)}$ and $K^{(2)}$ for the two layers, and the weights of penalty γ and β for must- and cannot-link constraints. Figure 11 reports the performance of GraphHAM on Facebook graph under different hyper-parameter settings. The model is generally more stable with respect to the number of groups, compared with the coefficients for the regularization term. Setting those hyper-parameters either too large or too small will compromise the performance. More specifically, performance peaks when $K^{(1)} = 12$ while $K^{(2)} = 5$ on the Facebook graph, which has 4 classes. Therefore, we believe a guidance for setting $K^{(2)}$ is to make it comparable with the number of classes while setting $K^{(1)}$ to be mildly larger than $K^{(2)}$. To balance the effect of the must- and cannot-link constraints, γ should be larger than β since the must-link set is usually much smaller.

3 Self-supervised Learning with Structural Property

To obtain more generalizable, transferable, and robust representations, the self-supervised learning (SSL) paradigm has recently emerged which enables graph neural networks (GNNs) to learn from pretext tasks constructed on unlabeled graph-structured data [65, 64, 200, 71]. As the current state-of-the-art SSL technique, graph contrastive learning (GCL) has attracted the most attention due to its simplicity and remarkable empirical performance [176, 205, 59, 198, 161, 169].

A typical GCL method works by creating augmented views of the input graph and learning node (or graph) representations by contrasting related graph objects against unrelated ones. Different contrastive objects are studied, such as node-node [205, 206, 133], node-(sub)graph [177, 59, 158] and graph-graph [11, 169, 161] contrastive pairs. The goal of GCL is to maximize the congruence between the representations of graph objects in augmented views, following the mutual information maximization (InfoMax) principle [60]. This makes graph augmentation one of the most critical designs in GCL, as it determines the effectiveness of the contrastive objective. However, despite various GCL methods have been proposed, it remains a mystery about what makes the most effective graph augmentations.

Unlike images, which can be augmented by rotation or cropping to naturally highlight the main subject from the background, it is less intuitive and more challenging to augment graphs due to the complicated topology structure of diverse nature (e.g., citation networks [152], social networks [125], chemical and biomedical molecules [96, 64]). Most existing works perform topology augmentations in a uniformly random manner [199, 205, 169, 11]. Although such a strategy indeed achieves a certain level of empirical success, it is far from optimal: recent studies show that perturbations on different edges post *unequal* influence on the graph spectrum [44, 22] while the uniformly random edge perturbation adopted in many GCL methods treats all edges equally and ignores such differences. Given the importance of graph spectrum for spectral filters in GNNs [39], such a discrepancy between uniform edge perturbations and their non-uniform influence on the graph spectrum urges us to rethink a fundamental but not yet clearly answered question: *Will graph spectrum based topology augmentations be more effective for GCL?*

In this work, we answer this question affirmatively. By studying the influence of different edge perturbation strategies on graph spectrum, we observe a clear positive relationship between the overall spectral change on augmented graphs and the resulting GCL performance. This empirical finding further motivates us to propose a principled Graph Contrastive Learning scheme with Topology Augmentation

guided by the Graph Spectrum, termed *GCL-TAGS*. Specifically, instead of perturbing edges uniformly at random, we search for graph augmentations that mostly change the graph spectrum of the input graph. By identifying sensitive edges where the graph spectrum is largely affected, GCL-TAGS allows the GNN encoder to focus on robust components (which can be hardly affected by small edge perturbations) in the spectral filters and to reduce its dependency on relatively vulnerable components (which can be easily affected). Therefore, the learned encoder captures the minimally sufficient information about the graph [171, 170] for the downstream tasks.

3.1 Related Work

Existing graph SSL methods focus on *self-prediction* [54, 134, 52, 200, 71] and *contrastive learning*. We focus on contrastive learning, and will mainly discuss existing designs of topology augmentation.

• **Graph Contrastive Learning.** Graph contrastive learning (GCL), also known as instance discrimination, leverages the InfoMax principle [60] to maximize the correspondence between related objects on the graph such that invariant property across objects is captured. Depending on how the related positive objects are defined, one line of work treats different parts of a graph as positive pairs, while constructing negative examples from a corrupted graph [64, 67, 177, 133, 158]. In such works, contrastive pairs can be defined as local nodes v.s. the entire graph [177], substructures v.s. graph [158], and the input graph v.s. reconstructed graph [133]. The other line of works exploits *graph augmentation* to generate multiple views, which enable more flexible contrastive pairs [169, 11, 199, 59, 137, 161, 198, 45]. By generating augmented views, the GNN model is encouraged to encode crucial graph information that is invariant to different views. While both topology and feature augmentations are explored in prior GCL works, we focus on topology augmentation strategies.

• **Graph Topology Augmentation.** The most widely adopted topology augmentation is the edge perturbation following *uniform distribution* [205, 169, 11, 199]. The underlying assumption is that each edge is equally important to the property of the input graph. However, a recent study shows that edge perturbations do not post equal influence to the graph spectrum [22] which summarizes a graph's structural property. To better preserve graph property that has been ignored by uniform perturbations, *domain knowledge* from network science is leveraged by considering the importance of edges measured via node centrality [206], the global diffusion matrix [59], and the random-walk based context graph [137]. While these works consider ad-hoc heuristics, our method targets the graph spectrum, which comprehensively summarizes global graph properties and plays a crucial role in the spectral filter of GNNs. To capture minimally sufficient information from the graph and remove redundancy that could compromise downstream performance, *adversarial training* strategy is paired with GCL for graph augmentation [161, 198, 45], following the information bottleneck (IB) [171] and InfoMin principle [170]. While the adversarial augmentation method requires frequent back-propagation during training, our method realizes a similar principle with a simpler but effective augmentation by maximizing the spectral change of graph with only one-time pre-computation.

3.2 Preliminaries

• **Notations.** We focus on connected undirected graphs $G = (\mathbf{X}, \mathbf{A})$ with n nodes and m edges, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ describes node features, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes its adjacency matrix such that $A_{ij} = 1$ if an edge exists between node i and j, otherwise $A_{ij} = 0$. The unnormalized Laplacian matrix of the graph is defined as $\mathbf{L}_{\text{u}} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_n)$ is the diagonal degree matrix with entry

$D_{ii} = \sum_{i=1}^{n} A_{ij}$ and $\mathbf{1}_n$ being an all-one vector with dimension n. The normalized Laplacian matrix is further defined as $\mathbf{L}_{\text{norm}} = \text{Lap}(\mathbf{A}) = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix.

• **Graph Spectrum.** By treating node features as signals, one can apply graph signal processing (GSP) techniques to conduct graph filtering for representation learning. The most essential component of GSP is the graph shift operator (GSO), which commonly adopts the normalized Laplacian matrix $\mathbf{L}_{\text{norm}}$ and admits an eigendecomposition as $\mathbf{L}_{\text{norm}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. The diagonal matrix $\mathbf{\Lambda} = \text{eig}(\mathbf{L}_{\text{norm}}) = \text{diag}(\lambda_1, \ldots, \lambda_n)$ consists of the real eigenvalues which are known as *graph spectrum*, and the corresponding $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n \times n}$ collecting the orthonormal eigenvectors are the *spectral bases*. Graph spectrum plays a significant role in analyzing and modeling graphs, as discussed in Appendix 1.1. On one hand, it comprehensively summarizes important graph structural properties, including connectivity [33], clusterability [89] and diffusion distance [55]. On the other hand, at the essence of many graph models, including GNNs, is the *spectral filter* which is defined on the graph spectrum, and different filters can manipulate graph signals in various ways, such as smoothing and denoising [150], abnormally detection [122] and clustering [179].

• **Graph Representation Learning.** Given a graph $G \in \mathcal{G}$, the goal of node representation learning is to train an encoder $f_\theta : \mathcal{G} \to \mathbb{R}^{n \times d'}$, such that $f_\theta(G)$ produces a low-dimensional vector for each node in G which can be served in downstream tasks, such as node classification. One can further obtain a graph representation by pooling the set of node representations via a readout function $g_\phi : \mathbb{R}^{n \times d'} \to \mathbb{R}^{d'}$, such that $g_\phi(f_\theta(G))$ outputs a low-dimensional vector for graph G which can be used in graph-level tasks such as graph classification or regression task.

• **Graph Contrastive Learning by Topology Augmentation.** GCL methods generally apply graph augmentation to perturb the input graph and decrease the amount of information inherited from the original graph; then they leverage the InfoMax principle [60] over the perturbed graph views such that an encoder is trained to capture the remaining information [161]. Given a graph $G \in \mathcal{G}$ with adjacency matrix $\mathbf{A}$, we denote a *topology augmentation scheme* as $T(\mathbf{A})$ and a sampled augmented view as $t(\mathbf{A}) \sim T(\mathbf{A})$. GCL with two-branch augmentation can be formulated as the following problem:

$$\text{GCL} : \min_{\Theta} \mathcal{L}_{\text{GCL}}(t_1(\mathbf{A}), t_2(\mathbf{A}), \Theta), \text{ s.t. } t_i(\mathbf{A}) \sim T_i(\mathbf{A}), i \in \{1, 2\} \tag{11}$$

where the contrastive loss $\mathcal{L}_{\text{GCL}}$ measures the disagreement between representations from contrastive positive pairs, which can be defined among different levels of representations, such as node-node [205], graph-graph [161], and node-graph [177, 59] representations. The topology augmentation scheme determines a distribution from which perturbed graphs are sampled in augmented views.

3.3 Methodology: Graph Contrastive Learning Guided by Graph Spectrum

In this section, we introduce our graph contrastive learning framework with topology augmentation guided by the change of graph spectrum (GCL-TAGS). We start with some empirical evidence on the relationship between the spectral change and GCL performance, then propose a new augmentation principle to maximize the spectral change when perturbing the graph.

3.3.1 Behavior of Edge Perturbation on Graph Spectrum Given that graph spectrum is a comprehensive manifestation of graph structural properties [33, 89, 55], to further understand its role in GCL, we study how the behavior of edge perturbation on graph spectrum correlates with GCL performance.

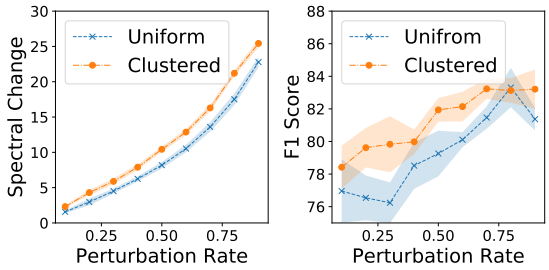


Fig. 13: Spectral change (LEFT) and downstream performance (RIGHT) of two augmentation schemes.

• **Pre-analysis Setup.** We take node representation learning on Cora dataset as an example, and adopt the same contrastive learning setup as in GRACE [205]. We consider two topology augmentation heuristics: 1) *uniform*: the original augmentation from GRACE [205] which removes edges with uniformly random probability; 2) *clustered*: a cluster-based strategy which removes edges within the same cluster with a larger probability. The cluster-based strategy explicitly modifies the connectivity between clusters which may result in large change on the graph spectrum as suggested by recent studies [30, 101], thus it stands in sharp contrast to the uniform perturbations. For the two augmentation branches, one is fixed with the uniform version, and we compare the strategies when the other augmentation branch adopts the uniform or the clustered perturbation. Figure 13 shows their comparison with respect to *spectral change* (measured by the L_2 distance of graph spectrum between the original and the augmented graphs), and *F1 score* (measuring the downstream node classification performance).

• **Remarks.** From Figure 13, we can clearly observe that under the same perturbation budget indicated by x-axis, the cluster-based strategy leads to a larger change on graph spectrum, while achieving better performance on the downstream task. This analysis suggests that an effective edge augmentation should pay more attention to sensitive edges that introduce large disturbance to graph spectrum. The performance gap between these two simple strategies suggests a distinct possibility to improve over the uniformly random augmentation. Unlike the proof-of-concept cluster-based heuristic, we aim on designing a principled topology augmentation by directly maximizing the spectral change.

3.3.2 Augmentation Scheme via Spectral Change Maximization

We now introduce our augmentation scheme guided by graph spectrum. We first define the edge perturbation based topology augmentation scheme determined by a Bernoulli *probability matrix*. Based on that, we formulate our augmentation principle as a spectral change maximization problem.

• **Edge Perturbation Based Augmentation Scheme.** We focus on topology augmentation using edge perturbation. Following the GCL formulation in Eq (11), we define topology augmentation $T(\mathbf{A})$ as a Bernoulli distribution $\mathcal{B}(\Delta_{ij})$ for each entry A_{ij}. All Bernoulli parameters for all entries constitute a *probability matrix* $\mathbf{\Delta} \in [0,1]^{n\times n}$. We can sample an edge perturbation matrix $\mathbf{E} \in \{0,1\}^{n\times n}$, where $E_{ij} \sim \mathcal{B}(\Delta_{ij})$ indicates whether to flip the edge between node i and j, and the edge is flipped if $E_{ij} = 1$ otherwise remaining unchanged. A sampled augmented graph is then obtained via:

$$t(\mathbf{A}) = \mathbf{A} + \mathbf{C} \circ \mathbf{E}, \ \mathbf{C} = \bar{\mathbf{A}} - \mathbf{A} \tag{12}$$

where $\bar{\mathbf{A}}$ is the complement matrix of the adjacency matrix $\mathbf{A}$, calculated by $\bar{\mathbf{A}} = \mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n - \mathbf{A}$, with $(\mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n)$ denoting the fully-connected graph without self-loops. Therefore, $\mathbf{C} = \bar{\mathbf{A}} - \mathbf{A} \in \{-1,1\}^{n\times n}$ denotes legitimate edge adding or removing operations for each node pair: edge adding

between node i and j is allowed if $C_{ij} = 1$, and edge removing is allowed if $C_{ij} = -1$. Taking the Hadamard product $\mathbf{C} \circ \mathbf{E}$ finally gives valid edge perturbations to the graph.

Since $\mathbf{E}$ is a matrix of random variables following Bernoulli distributions, we can easily obtain the *expectation* of sampled augmented graphs in Eq (12) as $\mathbb{E}[t(\mathbf{A})] = \mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}$. Therefore, the design of $\mathbf{\Delta}$ determines the topology augmentation scheme. Taking uniformly random edge removal as an example, the entry Δ_{ij} is set as a fixed dropout ratio if $C_{ij} = -1$; and 0 otherwise.

• **Spectral Change Maximization.** The default uniform edge perturbation adopted in many GCL methods is far from satisfactory. Motivated by our observation in Section 3.3.1, instead of setting fixed values for $\mathbf{\Delta}$, we propose to optimize it guided by graph spectrum. Specifically, we aim to search for $\mathbf{\Delta}$ that in expectation maximizes the spectral difference between the original graph and the augmented graph. Recall that we denote the normalized Laplacian matrix of $\mathbf{A}$ as $\text{Lap}(\mathbf{A})$, and the graph spectrum vector can be calculated by $\mathbf{\Lambda} = \text{eig}(\text{Lap}(\mathbf{A}))$. We formulate the following problem to search for the desired perturbation matrix $\mathbf{\Delta}$ in a single augmentation branch:

$$\text{Single-way scheme:} \quad \max_{\mathbf{\Delta} \in \mathcal{S}} \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta})) - \text{eig}(\text{Lap}(\mathbf{A}))\|_2^2 \tag{13}$$

where $\mathcal{S} = \{\mathbf{s} | \mathbf{s} \in [0, 1]^{n \times n}, \|\mathbf{s}\|_1 \leq \epsilon\}$ and ϵ controls the strength of graph perturbation. By solving Eq (13), we obtain the optimal Bernoulli probability matrix $\mathbf{\Delta}^*$, from which we can sample augmented views that in expectation differ the most from the original graph in graph spectrum. Note that Eq (13) only provides one augmented view, to further introduce flexibility for a two-branch augmentation framework and enlarge the spectral difference between the resulting two views, we extend Eq (13) as follows:

$$\text{Two-way scheme:} \quad \max_{\mathbf{\Delta}_1, \mathbf{\Delta}_2 \in \mathcal{S}} \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}_1)) - \text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}_2))\|_2^2 \tag{14}$$

where $\mathbf{\Delta}_i$ is the Bernoulli probability matrix for augmentation branch i's scheme $T_i(\mathbf{A})$ in Eq (11). Note that Eq (13) is a special case of Eq (14) when setting $\mathbf{\Delta}_2 = \mathbf{0}$. Eq (14) gives better flexibility yet also makes the nonconvex optimization problem harder to solve, thus we further simplify it by pushing two branches towards opposite directions: maximizing the spectral norm in one branch, while minimizing it in the other, which leads to the final objective for our augmentation scheme:

$$\text{Opposite-direction scheme:} \quad \max_{\mathbf{\Delta}_1 \in \mathcal{S}} \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_1), \quad \text{and} \quad \min_{\mathbf{\Delta}_2 \in \mathcal{S}} \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_2) \tag{15}$$

where $\mathcal{L}_{\text{GS}}(\mathbf{\Delta}) = \|\text{eig}(\text{Lap}(\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta}))\|_2^2$ measures the Graph Spectral norm under augmentation scheme with $\mathbf{\Delta}$. For scheme $T_1(\mathbf{A})$, $\mathbf{\Delta}_1$ produces views that overall have larger spectral norm than the original graph, while for $T_2(\mathbf{A})$, $\mathbf{\Delta}_2$ produces views with smaller spectrum. We can understand them as setting a spectral boundary for the input graph such that the encoder is trained to capture information that is essential and robust regarding perturbations within this region.

• **Optimizing $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$.** Eq (15) can be solved via projected gradient descent (for $\mathbf{\Delta}_2$) or ascent (for $\mathbf{\Delta}_1$). Taking $\mathbf{\Delta}_2$ as an example, its update works as follows:

$$\mathbf{\Delta}_2^{(t)} = \mathcal{P}_{\mathcal{S}}[\mathbf{\Delta}_2^{(t-1)} - \eta_t \nabla \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_2^{(t-1)})] \tag{16}$$

where t is the iteration step, $\eta_t > 0$ is the learning rate for step t, and $\mathcal{P}_{\mathcal{S}}(\mathbf{a}) = \text{argmin}_{\mathbf{s} \in \mathcal{S}} \|\mathbf{s} - \mathbf{a}\|_2^2$ is the projection operator at $\mathbf{a}$ over the constraint set $\mathcal{S}$. The calculation of gradient $\nabla \mathcal{L}_{\text{GS}}(\mathbf{\Delta}_2^{(t-1)})$ is done via chain rule. We now explain how to obtain a closed-form gradient over eigenvalues as it looks less straightforward. For a real and symmetric matrix $\mathbf{L}$, one can obtain the derivatives of its k-th eigenvalue λ_k by: $\partial \lambda_k / \partial \mathbf{L} = \mathbf{u}_k \mathbf{u}_k^\top$ [144], where $\mathbf{u}_k$ is the corresponding eigenvector. Note that the derivative calculation requires distinct eigenvalues, which does not hold for graphs satisfying *automorphism* [47].

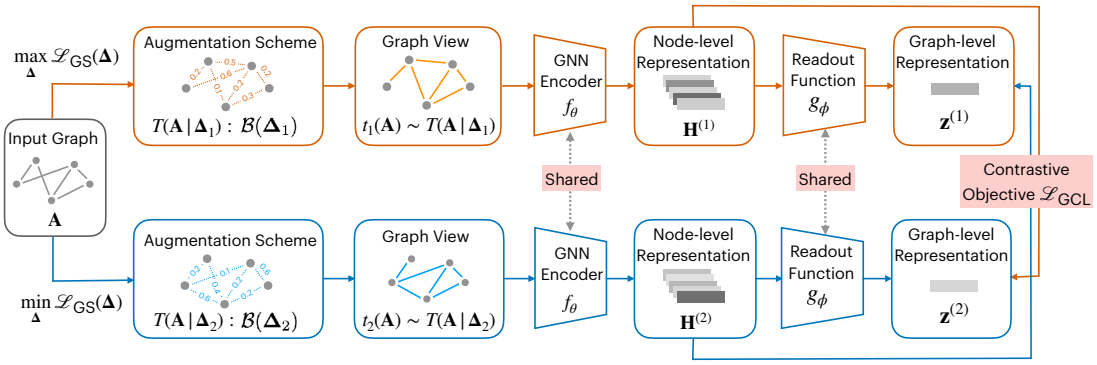


Fig. 14: The framework of GCL-TAGS contains topology augmentation and contrastive objective. The opposite-direction augmentation scheme guided by graph spectrum is pre-computed following Eq (15). The contrastive objective is to maximize the mutual information between node representations from one view and the graph representation from another view, and vice versa.

To avoid such cases, we add a small noise term to the adjacency matrix[4], e.g., $\mathbf{A} + \mathbf{C} \circ \mathbf{\Delta} + \varepsilon \times (\mathbf{N} + \mathbf{N}^\top)/2$, where each entry in $\mathbf{N}$ is sampled from a uniform distribution $\mathcal{U}(0, 1)$ and ε is a very small constant. Such a noise addition will almost surely break the graph automorphism, thus enabling a valid gradient calculation of eigenvalues.

For T iterations, the time complexity of optimizing the scheme is $\mathcal{O}(Tn^3)$ due to the eigen-decomposition $\text{eig}(\cdot)$ in $\mathcal{L}_{\text{GS}}$, which is prohibitively expensive for large graphs. To reduce the cost, instead of measuring the spectral change over all eigenvalues, we only maintain the K lowest- and highest-eigenvalues which are the most informative, as suggested by the spectral graph theory. The performance with different choices of K is studied in Appendix 1.4. Using selective eigen-decomposition via the Lanczos Algorithm [131], the time complexity of augmentation scheme optimization is recuded to $\mathcal{O}(TKn^2)$ [5].

3.3.3 Formulation and Framework of GCL-TAGS Figure 14 illustrates our GCL framework equipped with the spectrum-guided augmentation. We first pre-compute the Bernoulli probability matrix $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ by solving the optimization problem in Eq. Eq (15), which sets up the topology augmentation scheme. During contrastive learning, for each iteration, we sample two augmented graphs for the input graph $t_1(\mathbf{A}) \sim T(\mathbf{A}|\mathbf{\Delta}_1)$ and $t_2(\mathbf{A}) \sim T(\mathbf{A}|\mathbf{\Delta}_2)$. The augmented graphs are then fed into a GNN encoder f_θ, which outputs two sets of node representations $\mathbf{H}^{(1)}, \mathbf{H}^{(2)} \in \mathbb{R}^{n \times d'}$ corresponding to the two views. We then apply a graph pooling readout function g_ϕ to aggregate and transform the node representations and obtain graph representations $\mathbf{z}^{(1)}, \mathbf{z}^{(2)} \in \mathbb{R}^{d'}$. Finally, the GNN encoder and the readout function are trained by a contrastive objective $\mathcal{L}_{\text{GCL}}$ that maximizes the correspondence between local node representations of one view and the global graph representation of the other view. Given a set of training graphs $\mathcal{G}$, by putting all components together, we formulate GCL-TAGS as the following optimization problem:

$$\text{GCL-TAGS} : \min_{\theta,\phi} \mathcal{L}_{\text{GCL}}(t_1(\mathbf{A}), t_2(\mathbf{A}), \theta, \phi) = -\frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \left(\frac{1}{n} \sum_{i=1}^{n} \left(I(\mathbf{H}_i^{(1)}, \mathbf{z}^{(2)}) + I(\mathbf{H}_i^{(2)}, \mathbf{z}^{(1)}) \right) \right)$$
$$\text{s.t. } t_i(\mathbf{A}) \sim T(\mathbf{A}|\mathbf{\Delta}_i), i \in \{1, 2\}, \mathbf{\Delta}_1 = \text{argmax}_{\mathbf{\Delta} \in \mathcal{S}} \mathcal{L}_{\text{GS}}(\mathbf{\Delta}), \mathbf{\Delta}_2 = \text{argmin}_{\mathbf{\Delta} \in \mathcal{S}} \mathcal{L}_{\text{GS}}(\mathbf{\Delta}) \quad (17)$$

[4] The form of $(\mathbf{N} + \mathbf{N}^\top)/2$ is to keep the perturbed adjacency matrix symmetric for undirected graphs.
[5] Since we only require to precompute $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ once, the time complexity is totally acceptable.

Table 6: Node classification performance in *unsupervised* setting. The metric is *accuracy%*. **Bold** highlights that our method significantly outperforms baselines suggested by t-test with p-value$\leq$0.05.

	Dataset	Cora	Citeseer	PubMed	Wiki-CS	Amazon-Computer	Amazon-Photo	Coauthor-CS
	Raw-X	48.93±0.00	50.81±0.00	68.33±0.00	71.98±0.00	73.81±0.00	78.53±0.00	90.37±0.00
	S-GCN	81.34±0.35	70.42±0.45	79.82±0.41	77.19±0.12	86.51±0.54	92.42±0.16	93.03±0.31
	R-GCN	56.44±0.24	63.52±0.25	73.92±0.32	72.95±0.58	82.46±0.38	90.08±0.48	90.64±0.29
Baselines	GRACE [205]	83.33±0.43	72.10±0.54	78.72±0.13	80.14±0.48	89.53±0.35	92.78±0.30	91.12±0.20
	BGRL [169]	83.63±0.38	72.52±0.40	79.83±0.25	79.98±0.13	90.34±0.19	93.17±0.30	93.31±0.13
	GBT [11]	80.24±0.42	69.39±0.56	78.29±0.43	77.30±0.62	88.02±0.32	92.23±0.35	92.85±0.31
	MVGRL [59]	85.16±0.52	72.14±1.35	80.13±0.84	77.52±0.08	87.52±0.11	91.74±0.07	92.11±0.12
	GCA [206]	83.67±0.44	71.48±0.26	78.87±0.49	78.35±0.05	88.94±0.15	92.53±0.16	93.10±0.01
	GMI [133]	83.02±0.33	72.45±0.12	79.94±0.25	74.85±0.08	82.21±0.31	90.68±0.17	91.08±0.56
	DGI [177]	82.34±0.64	71.85±0.74	76.82±0.61	75.35±0.14	83.95±0.47	91.61±0.22	92.15±0.63
	GCL-TAGS	85.86±0.57	**72.76±0.63**	**81.54±0.24**	**82.13±0.15**	90.09±0.32	**93.52±0.26**	**93.91±0.24**

where $I(X_1, X_2)$ calculates the mutual information between variables X_1 and X_2, and we adopt InfoNCE as its estimator which is proven to be a lower bound of mutual information [173, 136]. Specifically, denoting cosine similarity as $\text{sim}(\cdot, \cdot)$, we estimate the mutual information as follows:

$$I(\mathbf{H}_i^{(a)}, \mathbf{z}^{(b)}) = \log \frac{\exp(\text{sim}(\mathbf{H}_i^{(a)}, \mathbf{z}^{(b)}))}{\sum_{j=1}^{n} \exp(\text{sim}(\widetilde{\mathbf{H}}_j, \mathbf{z}^{(b)}))} \tag{18}$$

where a and b index the augmented views, and $\widetilde{\mathbf{H}}$ is the node representations for a corrupted graph by randomly shuffling the features of the input graph [177, 59] to serve as negative examples. Note that the augmentation scheme is optimized prior to contrastive learning, which is a one-time computation thus does not introduce any extra complexity to the contrastive learning process.

3.4 Experiments

This section reports an extensive set of empirical evaluations of GCL-TAGS on a variety of graph datasets serving for downstream node classification, graph classification and regression tasks under unsupervised learning, transfer learning and adversarial attack settings.

• **Setup:** Our evaluation includes 25 graph datasets ranging from citation networks, social networks to chemical molecules. We compare GCL-TAGS against seven GCL baselines that serve for node representation learning and five baselines for graph representation learning. We adopt the following evaluation protocol for downstream tasks [161]: based on the representations given by the encoder, we train and evaluate a Logistic classifier or a Ridge regressor. We repeat all experiments for 10 times and report the mean and standard derivation of the evaluation metrics.

3.4.1 Unsupervised Learning Setting This setting is to evaluate the quality of learned graph/node representations. We apply different GCL methods for representation learning. A linear model is then trained using these learned representations as features for the downstream tasks and the resulting prediction performance is reported. We evaluate the effectiveness of GCL-TAGS for both node- and graph-level prediction tasks.

• **Node Classification Task.** The datasets include Cora, Citeseer, PubMed citation networks [152], Wiki-CS hyperlink network [119], Amazon-Computer/Photo co-purchase network [153], and Coauthor-CS network [153]. We compare GCL-TAGS against GCL methods that augment topology with uniformly

Table 7: Graph representation learning performance in *unsupervised* setting. TOP shows the biochemical and social network classification results on TU datasets (measured by *accuracy%*). BOTTOM shows the molecular regression (measured by *RMSE*) and classification (measured by *ROC-AUC%*) results on OGB datasets. **Bold** indicates that our method outperforms baselines with p-value≤ 0.05.

	Dataset	Biochemical Molecules				Social Networks				
		NCI1	PROTEINS	MUTAG	DD	COLLAB	RDT-B	RDT-M5K	IMDB-B	IMDB-M
	S-GIN	78.27±1.35	72.39±2.76	90.41±4.61	74.87±3.56	74.82±0.92	86.79±2.04	53.28±3.17	71.83±1.93	48.46±2.31
	R-GIN	62.98±0.10	69.03±0.33	87.61±0.39	74.22±0.30	63.08±0.10	58.97±0.13	27.52±0.61	51.86±0.33	32.81±0.57
Baselines	InfoGraph [158]	68.13±0.59	72.57±0.65	87.71±1.77	75.23±0.39	70.35±0.64	78.79±2.14	51.11±0.55	71.11±0.88	48.66±0.67
Baselines	GraphCL [199]	68.54±0.55	72.86±1.01	88.29±1.31	74.70±0.70	71.26±0.55	82.63±0.99	53.05±0.40	70.80±0.77	48.49±0.63
Baselines	MVGRL [59]	68.68±0.42	74.02±0.32	89.24±1.31	75.20±0.55	73.10±0.56	81.20±0.69	51.87±0.65	71.84±0.78	50.84±0.92
Baselines	AD-GCL [161]	69.67±0.51	73.59±0.65	89.25±1.45	74.49±0.52	73.32±0.61	85.52±0.79	53.00±0.82	71.57±1.01	49.04±0.53
Baselines	JOAO [198]	72.99±0.75	71.25±0.85	85.20±1.64	66.91±1.75	70.40±2.21	78.35±1.38	45.57±2.86	71.60±0.86	51.14±0.69
	GCL-TAGS	71.43±0.49	**75.78±0.41**	89.12±0.76	75.78±0.52	**75.01±0.45**	83.62±0.64	**54.10±0.49**	**73.65±0.69**	**52.16±0.72**

	Dataset	Regression (Metric: RMSE)			Classification (Metric: ROC-AUC%)				
		molesol	mollipo	molfreesolv	molbace	molbbbp	molclintox	moltox21	molsider
	S-GIN	1.173±0.057	0.757±0.018	2.755±0.349	72.97± 4.00	68.17±1.48	88.14±2.51	74.91±0.51	57.60±1.40
	R-GIN	1.706±0.180	1.075±0.022	7.526±2.119	75.07±2.23	64.48±2.46	72.29±4.15	71.53±0.74	62.29±1.12
Baselines	InfoGraph [158]	1.344±0.178	1.005±0.023	10.005±4.819	74.74±3.64	66.33±2.79	64.50±5.32	69.74±0.57	60.54±0.90
Baselines	GraphCL [199]	1.272±0.089	0.910±0.016	7.679±2.748	74.32±2.70	68.22±1.89	74.92±4.42	72.40±1.01	61.76±1.11
Baselines	MVGRL [59]	1.433±0.145	0.962±0.036	9.024±1.982	74.20±2.31	67.24±1.39	73.84±4.25	70.48±0.83	61.94±0.94
Baselines	AD-GCL [161]	1.217±0.087	0.842±0.028	5.150±0.624	76.37±2.03	68.24±1.47	80.77±3.92	71.42±0.73	63.19±0.95
Baselines	JOAO [198]	1.285±0.121	0.865±0.032	5.131±0.722	74.43±1.94	67.62±1.29	78.21±4.12	71.83±0.92	62.73±0.92
	GCL-TAGS	1.218±0.052	**0.802±0.019**	**4.531±0.463**	76.74±2.02	**69.59±1.34**	80.28±2.42	**72.83±0.62**	**64.87±0.88**

random edge perturbation (e.g., GRACE [205], BGRL [169], GBT [11]), centrality (GCA [206]), diffusion matrix (MVGRL [59]) and the original graph (e.g., GMI [133] and DGI [177]). We also consider a fully semi-supervised GCN (*S-GCN*), a randomly initialized untrained GCN (*R-GCN*) and using the raw node features as node representations (*Raw-X*). All the methods exploit a 2-layer GCN encoder and a downstream linear classifier with the same hyper-parameters for a fair comparison. We adopt random feature masking in GCL-TAGS, following the setup in SOTA works [206, 11]. We randomly split the datasets into training, validation and test set with ratio $10\%, 10\%, 80\%$ for evaluation purpose.

Table 6 shows that on the node classification task, GCL-TAGS achieves state-of-the-art performance in 6 out of the 7 datasets, 5 of which are significantly better than others. Specifically, comparing GCL-TAGS with MVGRL and GCA which use domain knowledge of the graph (e.g., node centrality or graph diffusion), the performance gain suggests the advantage of the spectrum based augmentation over previous domain-knowledge based heuristics. Meanwhile, GCL-TAGS is shown to be more effective than GRACE, BGRL and GBT which adopt uniformly random augmentation[6] and use a node-level contrastive objective. It is noteworthy that the representations learned by GCL methods achieve better performance than the semi-supervised model R-GCN, which suggests the effectiveness of self-supervised learning when label information is limited (e.g., only 10% data for training).

• **Graph Prediction Task.** We test on TU biochemical and social networks [125], Open Graph Benchmark (OGB) [63] and ZINC [64, 48] chemical molecules, and Protein-Protein Interaction (PPI) biological networks [64, 207] for graph prediction. We compare GCL-TAGS with five GCL methods including InfoGraph [158], GraphCL [199], MVGRL [59], AD-GCL (with fixed regularization weight) [161] and JOAO (v2) [198]. We use a 5-layer GIN encoder for all methods, including a semi-supervised S-GIN

[6] We also discuss the gain of spectrum augmentation in Appendix 1.4 by directly plugging our proposed augmentation into these frameworks.

Table 8: Graph classification performance in *transfer learning* setting on molecular classification task. The metric is *ROC-AUC%*. **Bold** indicates that our method outperforms baselines with p-value≤ 0.05.

Dataset	Pre-Train	ZINC-2M								PPI-306K
	Fine-Tune	BBBP	Tox21	SIDER	ClinTox	BACE	HIV	MUV	ToxCast	PPI
No-Pre-Train-GIN		65.8±4.5	74.0±0.8	57.3±1.6	58.0±4.4	70.1±5.4	75.3±1.9	71.8±2.5	63.4±0.6	64.8±1.0
Baselines	InfoGraph [158]	68.8±0.8	75.3±0.5	58.4±0.8	69.9±3.0	75.9±1.6	76.0±0.7	75.3±2.5	62.7±0.4	64.1±1.5
	GraphCL [199]	69.7±0.7	73.9±0.7	60.5±0.9	76.0±2.7	75.4±1.4	78.5±1.2	69.8±2.7	62.4±0.6	67.9±0.9
	MVGRL [59]	69.0±0.5	74.5±0.6	62.2±0.6	77.8±2.2	77.2±1.0	77.1±0.6	73.3±1.4	62.6±0.5	68.7±0.7
	AD-GCL [161]	70.0±1.1	76.5±0.8	63.3±0.8	79.8±3.5	78.5±0.8	78.3±1.0	72.3±1.6	63.1±0.7	68.8±1.3
	JOAO [198]	71.4±0.9	74.3±0.6	60.5±0.7	81.0±1.6	75.5±1.3	77.5±1.2	73.7±1.0	63.2±0.5	64.0±1.6
	GCL-TAGS	70.0±0.7	**78.0±0.5**	**64.7±0.5**	80.7±2.1	**79.9±0.7**	77.8±0.6	73.8±0.9	**64.2±0.4**	**70.0±0.8**

and a randomly initialized R-GIN. A readout function with a graph pooling layer and a 2-layer MLP is applied to generate graph representations. We adopt the given data split for OGB dataset, and use 10-fold cross validation for TU dataset as it does not provide such a split.

Table 7 summarizes the graph prediction performance. GCL-TAGS gives the best results on 13 out of 17 datasets, of which 10 are significantly better than others. Compared with GraphCL and JOAO which select the best combination of augmentations for each dataset from a pool of methods including edge perturbation, node dropping and subgraph sampling, GCL-TAGS using only edge perturbation based augmentation still outperforms them. This suggests the effectiveness of graph spectrum in guiding topology augmentation. Compared with MVGRL, our performance gain mainly comes from the augmentation scheme, as these two methods share similar contrastive objectives, and our augmentation guided by graph spectrum is clearly more effective than the widely adopted uniformly random augmentation. While AD-GCL and GCL-TAGS follow a similar principle to remove edges that carry non-important and redundant information, GCL-TAGS is more flexible since the augmentation scheme is optimized in an independent pre-computation step without interfering with the contrastive learning procedure.

3.4.2 Transfer Learning Setting This experiment evaluates the generalizability of the learned GNN models. Following [161, 64], we pre-train the GNN encoder on a large dataset using GCL methods, then fine-tune and evaluate it on other datasets. We focus on the graph classification task using chemical and biological datasets from [64]. The GCL baselines designed for graph representation learning are compared, as well as a reference model without pre-training (*No-Pre-Train-GIN*). Table 8 summarizes the performance. GCL-TAGS is shown to be more effective in learning generalizable encoders. This supports our augmentation principle: by perturbing edges that cause large spectral changes, the encoder is pre-trained to ignore unreliable structural information, such that the relationship between such information and downstream labels can be removed to mitigate the overfitting issue. The generalizability of the GNN encoder on molecule classification depends on the structural fingerprints such as *subgraphs* [43]. JOAO and GraphCL using *subgraph* sampling augmentation is outperformed by GCL-TAGS, which suggests that the graph spectrum could be another important fingerprint to study chemical and biological molecular properties.

3.4.3 Adversarial Attack Setting GNNs trained via GCL methods are shown to be more robust than encoders learned in a semi-supervised manner [53]. This setting focuses on further evaluating the robustness of different GCL methods when the input graphs are adversarially poisoned with perturbed topology. We adopt different poisoning attack strategies on the global graph structure, including *Random* (which randomly flips edges), *DICE* (which deletes edges internally and connects nodes externally across classes), *GF-Attack* (which maximizes a low-rank matrix approximation loss) and *Mettack* (which maximizes the training loss via meta-gradients). We test the perturbation ratios ranging from $\sigma \in \{0.05, 0.2\}$ for each attack strategy: $\sigma \times m$ edges are flipped for a graph with m edges.

Table 9: Node classification performance on Cora in *adversarial attack* setting (measured by *accuracy%*). **Bold** indicates that our method outperforms baselines with p-value≤ 0.05.

	Attack	Clean	Random		DICE		GF-Attack		Mettack	
	Ratio σ		0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2
	S-GCN	81.34±0.35	81.11±0.32	80.02±0.36	79.42±0.37	78.37±0.42	80.12±0.33	79.43±0.32	50.29±0.41	31.04±0.48
Baselines	GRACE [205]	83.33±0.43	83.23±0.38	82.57±0.48	81.28±0.39	80.72±0.44	82.59±0.35	80.23±0.38	67.42±0.59	55.26±0.53
	BGRL [169]	83.63±0.38	83.12±0.34	83.02±0.39	82.83±0.48	81.92±0.39	82.10±0.37	80.98±0.42	70.23±0.48	60.42±0.54
	GBT [11]	80.24±0.42	80.53±0.39	80.20±0.35	80.32±0.32	80.20±0.34	79.89±0.41	78.25±0.49	63.26±0.69	53.89±0.55
	MVGRL [59]	85.16±0.52	85.28±0.49	84.21±0.42	83.78±0.35	83.02±0.40	83.79±0.39	82.46±0.52	73.43±0.53	61.49±0.56
	GCA [206]	83.67±0.44	83.33±0.46	82.49±0.37	82.20±0.32	81.82±0.45	81.83±0.36	79.89±0.47	58.25±0.68	49.25±0.62
	GMI [133]	83.02±0.33	83.14±0.38	82.12±0.44	82.42±0.44	81.13±0.49	82.13±0.39	80.26±0.48	60.59±0.54	53.67±0.68
	DGI [177]	82.34±0.64	82.10±0.58	81.03±0.52	80.48±0.38	79.89±0.43	81.30±0.54	79.88±0.58	71.42±0.63	63.93±0.58
	GCL-TAGS	85.86±0.57	**86.29±0.52**	**86.21±0.78**	**85.52±0.59**	**84.30±0.63**	**85.08±0.77**	**84.28±0.82**	**77.28±0.82**	**69.92±0.83**

Table 9 reports the node classification performance under adversarial attack. The encoders learned by GCL methods with graph augmentations are generally more robust to perturbed graph structure compared with S-GCN. GCL-TAGS outperforms baselines with a clear margin, even under the strong Mettack which solicits the downstream label information. This shows the advantage of our augmentation scheme by spectral change maximization: GCL-TAGS enables the encoder to stay invariant to the adversarially perturbed graph if its spectrum falls into the range captured by the opposite-direction augmentation scheme.

4 Conclusion

In this chapter, we demonstrate the possibility of improving unsupervised machine learning with graph structural information, including explicit pairwise link, implicit cluster and global structural property summarized by graph spectrum. Firstly, we treat the explicit edge as an indication of entity similarity. For the task of user representation learning, different modalities of user-generated data, i.e., social network and user reviews, are jointly modeled via a generative model to integrate user representation learning with network and topic embedding. The learned user representations are interpretable and predictive, indicated by the performance improvement in many important tasks such as link prediction and expert recommendation. In the second work, we further capture a hierarchy of implicit node groups and the nodes' affiliation to such groups. We align the group hierarchy with the convolutional layers in GCNs and design a joint node-level and group-level attention mechanism to aggregate the neighbor structure more accurately. Finally, we explore on improving an advanced self-supervised learning paradigm using graph structural property, that is, contrastive learning. Instead of generating graph augmentation using uniformly random edge perturbation, we propose a guided augmentation scheme by maximizing the change of graph spectrum. These works show our successful trials in harnessing graph structure for unsupervised learning from different perspectives.

While we enjoy the opportunities brought by graph structure to facilitate machine learning, potential threats also arise due to the complicated dependency structure among entities. Such dependency may inevitably expose vulnerability to adversary or amplify unwanted bias in the dataset, threatening robustness and fairness of the learning models. In the next chapter, we will introduce our efforts in understanding and mitigating structural threats such that more trustworthy machine learning solutions can be achieved.

Chapter 3

Understanding Graph Structural Threats in Machine Learning

While we take advantage of graph structure to advance machine learning, potential threats could arise when the graph structure is used in an undesired way. The misuse of graph structure could lead to serious threats to the accountability and trustworthiness of machine learning models. The graph structure driven by homophily and social influence is inevitably affected by sensitive attributes of entities, e.g., people with the same skin color tend to connect, and a loan or job recommender system learned on such structure may favor or disregard groups. The structure can inherit and even magnify undesired social discrimination, which raises *fairness* issues and bias concerns. Meanwhile, the graph structure exposing dependency information gives malicious attackers more room to break in, e.g., phishing users can deceive recommender systems and worm normal users' trust by strategically making connections with their friends. This raises *robustness* issues and security concerns. These issues indicate that introducing graph structure in machine learning could be risky, because biased or perturbed graph structure could greatly mislead the machine learning models. This calls for careful treatment of graph structure when dealing with real-world applications, such as job recommendations or friend recommendations.

In this chapter, we introduce our efforts in understanding potential fairness and robustness issues raised by biased or perturbed graph structure. In Section 1, we propose a principled unbiased graph embedding framework to cope with the ethical issues when a machine learning model is trained on a biased structure [184]. Such biased models could output unfair decision making to different groups of users based on their sensitive demographic features, and simply removing sensitive features cannot amend the biased structure. By studying the graph generation process, we aim to learn node embeddings from an *underlying bias-free graph* whose edges are generated without any influence from sensitive attributes. Two alternative solutions are further provided to uncover the bias-free graph from the given observed graph. In Section 2, we design a new structural attack method in the Fourier domain to understand the robustness issue when the graph structure used to train a machine learning model can be penetrated by an adversary [102]. Specifically, we define *spectral distance* between the original and perturbed graph and attack the graph structure by directly maximizing the spectral distance. This work expands the scope of verifying and enhancing graph embedding models' robustness in both spatial and Fourier domains.

1 Fair Machine Learning with Biased Structure

The observed connections in a graph are inevitably affected by certain sensitive attributes (such as gender and age of a user in a social network), which should be withheld from the downstream tasks [135]. Without proper intervention, the learnt node embeddings can inherit undesired sensitive information that can lead to bias or fairness concerns when used in downstream tasks [138, 15]. For example, in a social

network, if the users with the same gender tend to connect more often, the learnt embeddings can inherit such gender information and lead to gender bias by only recommending friends to a user with the same gender identity. And from the data privacy perspective, this also opens up the possibility for extraction attacks from the learnt node embeddings [159].

There is rich literature in enforcing unbiasedness/fairness in algorithmic decision making, especially in classical classification problems [76, 201, 31]. Unbiased graph embedding has just started to attract research attentions in recent years. To date, the most popular recipe for unbiased graph embedding is to add adversarial regularizations to the loss function, such that the sensitive attributes cannot be predicted by the learnt embeddings [112, 15, 4, 37]. For example, making a discriminator built on the node embeddings fail to predict the sensitive attributes of the nodes. However, such a regularization is only a necessary condition for unbiased node embeddings, and it usually hurts the utility of the embeddings in downstream tasks (a trivial satisfying solution is to randomize the embeddings). Besides these regularization-based solutions, Fairwalk [138] modifies the random walk strategy in the node2vec algorithm [52] into two levels: when choosing the next node on a path, it first randomly selects a group defined by sensitive attributes, and then randomly samples a reachable node from that group. DeBayes [19] proposes to capture the sensitive information by a prior function in Conditional Network Embedding [77], such that the learned embeddings will not carry the sensitive information. Nevertheless, both Fairwalk and DeBayes are based on specific graph embedding methods; and how to generalize them to other types of graph embedding methods such as GAT [175] or SGC [190] is not obvious.

Moving beyond the existing unbiased graph embedding paradigm, in this paper, we propose a principled new framework for the purpose with theoretical justifications. Our solution is to learn node embeddings from an *underlying bias-free graph* whose edges are generated without any influence from sensitive attributes. Specifically, as suggested by Pfeiffer et al. [135], the generation of a graph can be treated as a two-phase procedure. In the first phase, the nodes are connected with each other solely based on global graph *structural properties*, such as degree distributions, diameter, edge connectivity, clustering coefficients and etc., resulting in an *underlying structural graph*, free of influences from node attributes. In the second phase, the connections are *re-routed* by the node attributes (including both sensitive and non-sensitive attributes). For example, in a social network, users in the same age group tend to be more connected than those in different age groups, leading to the final observed graph biased by the age attribute. Hence, our debiasing solution is to filter out the influence from sensitive attributes on the underlying structural graph to create a bias-free graph (that only has non-sensitive attributes or no attributes) from the observed graph, and then perform embedding learning on the bias-free graph.

We propose two alternative ways to uncover the bias-free graph from the given graph for learning node embeddings. The first is a weighting-based method, which reweighs the graph reconstruction based loss function with importance sampling on each edge, such that in expectation the derived loss is as calculated on the bias-free graph. This forms a sufficient condition for learning unbiased node embeddings: when the reconstruction loss is indeed defined on the corresponding bias-free graph, the resulting node embeddings are unbiased, since the bias-free graph is independent from the sensitive attributes. The second way is via regularization, in which we require the probabilities of generating an edge between two nodes with and without the sensitive attributes to be the same. This forms a necessary condition for learning unbiased node embeddings: when the learning happens on the bias-free graph, the learnt embeddings should not differentiate if any sensitive attributes participated in the generation of observed graph, i.e., the predicted edge generation should be independent from the sensitive attributes. These two methods are complementary and can be combined to control the trade-off between utility and unbiasedness.

1.1 Related Work

• **Graph Embedding**. Graph embedding aims to map graph nodes to low-dimensional vector representations such that the original graph can be reconstructed from these embeddings. Traditional approaches include matrix factorization and spectral clustering techniques [126, 7]. Recent years have witnessed numerous successful advances in deep neural architectures for learning graph embeddings. Deepwalk [134] and node2vec [52] utilize a skip-gram [120] based objective to recover the node context in random walks on a graph. Graph Convolutional Networks (GCNs) learn a node's embedding by aggregating the features from its neighbors supervised by node/edge labels in an end-to-end manner. These techniques are widely applied in various real-world applications, such as friend recommendation in social network [105], content recommendation [192], protein structure prediction [74] and many more.

• **Unbiased and Fair Node Embeddings**. In real-life scenarios, sensitive attributes such as age, gender, skin color, religion, and region are inevitably involved in the generation and evolution of graph data, which raises severe bias and fairness concern when applying graph embedding techniques for real-world tasks. For example, people with similar age tend to connect closer; thus, graph embedding models built on such input for downstream applications, such as loan application and job recommendation, may unintentionally favor or disregard one group, causing biased and unfair treatments. This realistic and ethical concern sets a higher bar for the graph embedding models to learn effective and unbiased embeddings.

Recent efforts on unbiased and fair graph embedding mainly focus on *pre-processing*, *algorithmic* and *post-processing* steps in the learning pipeline. The *pre-processing* solutions modify the training data to reduce the leakage of sensitive attributes [20]. Fairwalk [138] is a typical pre-processing method which modifies the sampling process of random walk on graphs by giving each group of neighbors an equal chance to be chosen. However, such pre-processing may shift the data distribution and leads the trained model to inferior accuracy measured on test set. The *post-processing* methods employ discriminators to correct the learnt embeddings to satisfy specific fairness constraints [57]. However, such ad-hoc post-correction is detached from model training which may heavily degrade model's prediction quality.

Our work falls into the category of *algorithmic* methods, which modify the learning objective to prevent bias from the node embeddings. The most popular algorithmic solution is adding (adversarial) regularizations as constraints to filter out sensitive information [15, 36, 4]. Compositional fairness constraints [15] are realized by a composition of discriminators for a set of sensitive attributes jointly trained with the graph embedding model. Similarly, FairGNN [36] adopts a fair discriminator but focuses on debiasing with missing sensitive attribute values. Different from regularization based methods. DeBayes [19] reformulates the maximum likelihood estimation with a biased prior which absorbs the information about sensitive attributes; but this solution is heavily coupled with the specific embedding method thus is hard to generalize. Our method differs from these previous works by learning embeddings from an underlying bias-free graph. We investigate the generation of the given graph and remove the influence from sensitive attributes in the generative process to uncover a bias-free graph for graph embedding.

• **Generative Graph Models**. *Generative graph models* [5, 135] focus on the statistical process of graph generation by modeling the joint distributions of edges conditioned on node attributes and graph structure. For example, Attributed Graph Model (AGM) [135] jointly models graph structure and node attributes in a two step graph generation process. It first exploits a generative graph model to compute underlying structural edge probabilities based on the structural properties of an observed graph. AGM then learns attribute correlations among edges from the observed graph and combines them with the structural probabilities to sample graphs conditioned on attribute values, while keeping the expected edge probabilities and degrees of the input structural graph model. This process motivates us to uncover an underlying bias-free graph by separating out sensitive attributes and only conditioning on non-sensitive attributes for calculating edge probabilities.

1.2 Preliminaries

In this section, we first introduce our notations and general graph embedding concepts. Since the bias and fairness issues emerge most notably in prediction tasks involving humans, such as loan application [86] or criminal justice [9], we will use user-related graphs as running examples to discuss our criterion for unbiased graph embedding. But we have to emphasize this setting is only to illustrate the concept of unbiased graph embedding; and our proposed solution can be applied to any graph data.

• **Notations**. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ be an undirected, attributed graph with a set of N nodes $\mathcal{V}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a set of N attribute vectors $\mathcal{A}$ (one attribute vector for each node). We use (u, v) to denote an edge between node $u \in \mathcal{V}$ and node $v \in \mathcal{V}$. The number of attributes on each node is K, and $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N\}$, where $\boldsymbol{a}_u$ is a K-dimensional attribute vector for node u. Without loss of generality, we assume only the first m attributes are sensitive, and $a_u[:m]$ and $a_u[m:]$ stands for the first m sensitive attributes and the rest of the attributes that are non-sensitive, respectively. We assume all attributes are categorical and $\mathcal{S}_i$ is the set of all possible values for attribute i. For example, if node u is a user node, and the i-th attribute is gender with possible values $\mathcal{S}_i = \{\textit{Female}, \textit{Male}, \textit{Unknown}\}$, then $\boldsymbol{a}_u[i] = \textit{Female}$ indicates u is a female.

In the problem of graph embedding learning, we aim to learn an encoder $\mathrm{ENC} : \mathcal{V} \to \mathbb{R}^d$ that maps each node u to a d-dimensional embedding vector $\boldsymbol{z}_u = \mathrm{ENC}(u)$. We focus on the *unsupervised* embedding setting which does not require node labels and the embeddings are learned via the *link prediction* task. In this task, a scoring function $\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ with parameters $\boldsymbol{\theta}$ is defined to predict the probability of an edge between node u and node v in the given graph, i.e., $(u, v) \in \mathcal{E}$. The loss for learning node embeddings and parameters of the encoder and scoring function are defined by:

$$\sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \tag{19}$$

where $\mathcal{L}_{edge}$ is a per-edge loss function on $(u, v) \in \mathcal{E}$. Such loss functions generally aim to maximize the likelihood of observed edges in the given graph, comparing to the negative samples of node pairs where edges are not observed [121, 52].

• **Unbiased Graph Embedding**. Given a user node u, we consider its embedding $\boldsymbol{z}_u$ as unbiased with respect to an attribute i if it is independent from the attribute, $\boldsymbol{z}_u \perp \boldsymbol{a}_u[i], u \in \mathcal{V}$. Prior work measures such unbiasedness in the learnt embeddings by their ability to predict the value of the sensitive attributes [15, 129, 19]. For example, such work evaluates the unbiasedness of learnt embeddings by training a classifier on a subset of nodes' embeddings and their sensitive attributes. If the classifier cannot predict the sensitive attribute values of nodes in the test set, one claims that the embeddings have low bias. If the prediction accuracy equals to that from random embeddings, the learnt embeddings are considered bias-free. In fact, such classifiers are often used as discriminators in adversarial methods where the classifier and the embeddings are learnt jointly: the embeddings are pushed in directions where the classifier has low prediction accuracy [112, 15].

There are also studies that use fairness measures such as demographic parity or equalized opportunity to define the unbiasedness of learnt embeddings [57, 19]. But such fairness measures can only evaluate the fairness of the final prediction results for the intended downstream tasks, but cannot assess whether the embeddings are biased by, or contain any information about, sensitive attributes. In particular, fairness in a downstream task is only a necessary condition for unbiased embedding learning, not sufficient. The logic is obvious: unbiased embeddings can lead to fair prediction results as no sensitive attribute information is involved; but obtaining fairness in one task does not suggest the embeddings themselves are unbiased, e.g., those embeddings can still lead to unfair results in other tasks or even the fair results are obtained by other means, such as post-processing of the prediction results [189].

1.3 Effect of Attributes in Graph Generation

We now discuss the generation of an observed graph by explicitly modeling the effects of node attributes in the process. In particular, we assume that there is an underlying *structural graph* behind an observed graph, whose edge distribution is governed by the global graph properties such as degree distributions, diameter, and clustering coefficients. Then the attributes in $\mathcal{A}$ will modify the structural edge distribution based on effects like *homophily* in social networks, where links are rewired based on the attribute similarities of the individuals [117, 87]. The new distribution is then used to generate the observed graph.

Formally, let $\mathcal{M}$ be a generative graph model and Θ_M be the set of parameters that describe properties of the underlying structural graph. In particular, this set of parameters Θ_M is independent from node attributes in $\mathcal{A}$. We consider the class of generative graph models that represent the set of possible edges in the graph as binary random variables $E_{uv}, u \in \mathcal{V}, v \in \mathcal{V}$: i.e., the event $E_{uv} = 1$ indicates $(u, v) \in \mathcal{E}$. The model $\mathcal{M}$ assigns a probability to E_{uv} based on Θ_M, $P_M(E_{uv} = 1|\Theta_M)$. Therefore, the edge set $\mathcal{E}$ can be considered as samples from $Bernoulli(P_M(E_{uv} = 1|\Theta_M))$. There are many such structural models $\mathcal{M}$ such as the Chung Lu model [32] and Kronecker Product Graph Model [91]. Note that $\mathcal{M}$ does not consider attributes in $\mathcal{A}$ in the generation of the graph.

Now we involve the attributes in the generation of a graph. Specifically, let $C_{uv} \in \{(\boldsymbol{a}_i, \boldsymbol{a}_j)|i \in \mathcal{V}, j \in \mathcal{V}\}$ be a random variable indicating the *attribute value combination* of a pair of nodes u and v, which is independent from Θ_M. We should note that C_{uv} on different node pairs (u, v) can be the same, as many different node pairs can share the same attribute value combination. $P_o(E_{uv} = 1|C_{uv} = \boldsymbol{a}_{uv}, \Theta_M)$ is the conditional probability of an edge given the corresponding attributes on the incident nodes and structural parameters Θ_M, where $\boldsymbol{a}_{uv} = (\boldsymbol{a}_u, \boldsymbol{a}_v)$ denotes the observed attribute value combination on nodes u and v. Then based on Bayes' Theorem, we have

$$\begin{aligned} P_o(E_{uv} = 1|C_{uv} = \boldsymbol{a}_{uv}, \Theta_M) &= \frac{P_o(C_{uv} = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M)P_o(E_{uv} = 1|\Theta_M)}{P_o(C_{uv} = \boldsymbol{a}_{uv}|\Theta_M)} \\ &= P_M(E_{uv} = 1|\Theta_M)\frac{P_o(C_{uv} = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M)}{P_o(C_{uv} = \boldsymbol{a}_{uv}|\Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V} \end{aligned} \tag{20}$$

where the prior distribution on E_{uv} is specified by the structural model $\mathcal{M}$: i.e., $P_o(E_{uv} = 1|\Theta_M) = P_M(E_{uv} = 1|\Theta_M)$, and the posterior distribution accounts for the influences from the attribute value combinations. Therefore, the edge distribution used to generate the observed graph with node attributes is a modification of an structural graph distribution defined by $\mathcal{M}$ and Θ_M. It is important to note that in the generative process, the node attributes are given ahead of graph generation. They are the input to the generation model, not the output. Hence $P_o(C_{uv} = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M)$ is the same for all edges whose incident nodes have the attribute combination C_{uv}, no matter where they are, since C_{uv} is independent from the graph structure Θ_M.

To simplify the notation, let us define a function that maps the attribute value combination $\boldsymbol{a}_{uv}$ to the probability ratio that modifies the structural graph into the observed graph by

$$R(\boldsymbol{a}_{uv}) \coloneqq \frac{P_o(C_{uv} = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M)}{P_o(C_{uv} = \boldsymbol{a}_{uv}|\Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V}$$

Thus we can rewrite Eq (20) by

$$P_o(E_{uv} = 1|C_{uv} = \boldsymbol{a}_{uv}, \Theta_M) = P_M(E_{uv} = 1|\Theta_M)R(\boldsymbol{a}_{uv}) \tag{21}$$

In this way, we explicitly model the effect of node attributes by $R(\boldsymbol{a}_{uv})$, which modifies the structural graph distribution $P_M(E_{uv} = 1|\Theta_M)$ for generating the observed graph $\mathcal{G}$.

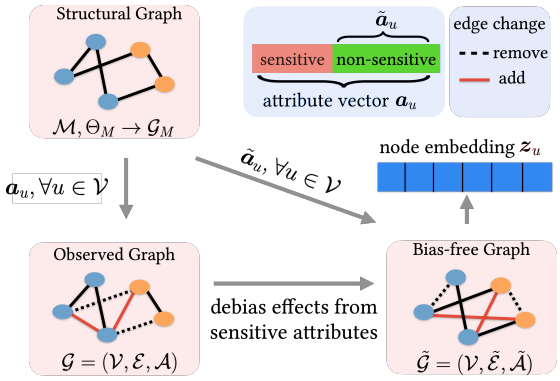


Fig. 15: Illustration of UGE. The color of nodes represents the value of sensitive attributes, and different line styles suggest how the observed edges are influenced by sensitive attributes in the generative process.

1.4 Methodology: Unbiased Graph Embedding from a Bias-Free Graph

In a nutshell, we aim to get rid of sensitive attributes and modify the structural edge distribution by only conditioning on non-sensitive attributes. This gives us the edge distribution of a bias-free graph, from which we can learn unbiased node embedding. Consider a world without the sensitive attributes, and the attribute vector of node u becomes $\widetilde{\boldsymbol{a}}_u = \boldsymbol{a}_u[m:]$, which only include non-sensitive attributes in $\boldsymbol{a}_u$. We denote $\widetilde{\mathcal{G}} = (\mathcal{V}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{A}})$ as the corresponding new graph generated with $\widetilde{\boldsymbol{a}}_u, \forall u \in \mathcal{V}$, and $\widetilde{\boldsymbol{a}}_{uv} = (\widetilde{\boldsymbol{a}}_u, \widetilde{\boldsymbol{a}}_v)$. Therefore, $\widetilde{\mathcal{G}}$ is a bias-free graph without influence from sensitive attributes. If we can learn node embeddings from $\widetilde{\mathcal{G}}$ instead of $\mathcal{G}$, the embeddings are guaranteed to be unbiased with respect to sensitive attributes. Specifically, the edge probabilities used for generating $\widetilde{\mathcal{G}}$ can be written as

$$P_{\widetilde{o}}(E_{uv} = 1 | \widetilde{C}_{uv} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M) = P_M(E_{uv} = 1 | \Theta_M) \widetilde{R}(\widetilde{\boldsymbol{a}}_{uv}), \tag{22}$$

where

$$\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv}) := \frac{P_{\widetilde{o}}(\widetilde{C}_{uv} = \widetilde{\boldsymbol{a}}_{uv} | E_{uv} = 1, \Theta_M)}{P_{\widetilde{o}}(\widetilde{C}_{uv} = \widetilde{\boldsymbol{a}}_{uv} | \Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V}, \tag{23}$$

$\widetilde{C}_{uv} \in \{(\widetilde{\boldsymbol{a}}_i, \widetilde{\boldsymbol{a}}_j) | i \in \mathcal{V}, j \in \mathcal{V}\}$ is the random variable indicating attribute value combinations without sensitive attributes, and $P_{\widetilde{o}}$ indicates the distribution used in generating $\widetilde{\mathcal{G}}$. We name the methods that learn embeddings from $\widetilde{\mathcal{G}}$ as UGE, simply for **U**nbiased **G**raph **E**mbedding. We illustrate the general principle of UGE in Figure 15. Next we discuss two instances of UGE. The first is called UGE-W, which reweighs the per-edge loss such that the total loss is from $\widetilde{\mathcal{G}}$ in expectation. The second is called UGE-R, which adds a regularization term to satisfy the properties as embeddings directly learnt from $\widetilde{\mathcal{G}}$.

1.4.1 Weighting-Based UGE

We modify the loss function in Eq (19) by reweighing the loss as

$$\mathcal{L}_{UGE-W}(\mathcal{G}) = \sum_{(u,v) \in \mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})}. \tag{24}$$

The following theorem shows that in expectation Eq (24) is equivalent as learning embeddings from $\widetilde{\mathcal{G}}$.

Theorem 1. *Given a graph $\mathcal{G}$, and $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})/R(\boldsymbol{a}_{uv}), \forall(u,v) \in \mathcal{E}$, $\mathcal{L}_{UGE-W}(\mathcal{G})$ is an unbiased loss with respect to $\widetilde{\mathcal{G}}$.*

Proof. We take expectation over the edge observations in $\mathcal{G}$ as

$$\begin{aligned}
\mathbb{E}\big[\mathcal{L}_{UGE-W}(\mathcal{G})\big] &= \mathbb{E}\left[\sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u,\boldsymbol{z}_v))\frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})}\right] \\
&= \mathbb{E}\left[\sum_{u\in\mathcal{V},v\in\mathcal{V}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u,\boldsymbol{z}_v))\frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})}\cdot E_{uv}\right] \\
&= \sum_{u\in\mathcal{V},v\in\mathcal{V}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u,\boldsymbol{z}_v))\frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})}\cdot P_o(E_{uv}=1|C_{uv}=\boldsymbol{a}_{uv},\Theta_M) \\
*&= \sum_{u\in\mathcal{V},v\in\mathcal{V}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u,\boldsymbol{z}_v))\cdot P_{\widetilde{o}}(E_{uv}=1|\widetilde{C}_{uv}=\widetilde{\boldsymbol{a}}_{uv},\Theta_M) \\
&= \mathbb{E}\left[\sum_{(u,v)\in\widetilde{\mathcal{E}}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u,\boldsymbol{z}_v))\right].
\end{aligned} \tag{25}$$

The step marked by $*$ uses Eq (21) and Eq (22).

UGE-W is related to the idea of importance sampling [83], which analyzes the edge distribution of the bias-free graph $\widetilde{\mathcal{G}}$ by observations from the given graph $\mathcal{G}$. The only thing needed for deploying UGE-W in existing graph embedding methods is to calculate the weights $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})/R(\boldsymbol{a}_{uv})$. To estimate $R(\boldsymbol{a}_{uv})$, we need estimates for two probabilities $P_o(C_{uv}=\boldsymbol{a}_{uv}|E_{uv}=1,\Theta_M)$ and $P_o(C_{uv}=\boldsymbol{a}_{uv}|\Theta_M)$. With maximum likelihood estimates on the observed graph, we have

$$P_o(C_{uv}=\boldsymbol{a}_{uv}|E_{uv}=1,\Theta_M) \approx \frac{\sum_{(i,j)\in\mathcal{E}}\mathbb{I}[\boldsymbol{a}_{ij}=\boldsymbol{a}_{uv}]}{|\mathcal{E}|}, \tag{26}$$

$$P_o(C_{uv}=\boldsymbol{a}_{uv}|\Theta_M) \approx \frac{\sum_{i\in\mathcal{V},j\in\mathcal{V}}\mathbb{I}[\boldsymbol{a}_{ij}=\boldsymbol{a}_{uv}]}{N^2}. \tag{27}$$

Similarly we can estimate $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})$ by

$$P_o(\widetilde{C}_{uv}=\widetilde{\boldsymbol{a}}_{uv}|E_{uv}=1,\Theta_M) \approx \frac{\sum_{(i,j)\in\widetilde{\mathcal{E}}}\mathbb{I}[\widetilde{\boldsymbol{a}}_{ij}=\widetilde{\boldsymbol{a}}_{uv}]}{|\widetilde{\mathcal{E}}|}, \tag{28}$$

$$P_o(\widetilde{C}_{uv}=\widetilde{\boldsymbol{a}}_{uv}|\Theta_M) \approx \frac{\sum_{i\in\mathcal{V},j\in\mathcal{V}}\mathbb{I}[\widetilde{\boldsymbol{a}}_{ij}=\widetilde{\boldsymbol{a}}_{uv}]}{N^2}. \tag{29}$$

Note that the estimation of $P_o(\widetilde{C}_{uv}=\widetilde{\boldsymbol{a}}_{uv}|E_{uv}=1,\Theta_M)$ is based on $\widetilde{\mathcal{E}}$, which is unfortunately unobservable. But we can approximate $P_o(\widetilde{C}_{uv}=\widetilde{\boldsymbol{a}}_{uv}|E_{uv}=1,\Theta_M)$ with $\mathcal{E}$ in the following way: after grouping node pairs by non-sensitive attribute value combinations $\widetilde{\boldsymbol{a}}_{uv}$, the sensitive attributes only re-

route the edges but do not change the number of edges in each group, and thus we have

$$
\begin{aligned}
P_o(\widetilde{C}_{uv} = \widetilde{\boldsymbol{a}}_{uv} | E_{uv} = 1, \Theta_M) &\approx \frac{\sum_{(i,j)\in\widetilde{\mathcal{E}}} \mathbb{I}[\widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}]}{|\widetilde{\mathcal{E}}|} \\
&= \frac{\sum_{i\in\mathcal{V}, j\in\mathcal{V}, \widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}} \mathbb{I}[(i,j) \in \widetilde{\mathcal{E}}]}{|\widetilde{\mathcal{E}}|} \\
&= \frac{\sum_{i\in\mathcal{V}, j\in\mathcal{V}, \widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}} \mathbb{I}[(i,j) \in \mathcal{E}]}{|\widetilde{\mathcal{E}}|} \\
&= \frac{\sum_{(i,j)\in\mathcal{E}} \mathbb{I}[\widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}]}{|\mathcal{E}|}.
\end{aligned} \tag{30}
$$

For node pairs with the same attribute value combination, Eq (26)-Eq (29) only need to be calculated once instead of for each pair. This can be done by first grouping the pairs by their attribute value combinations and then perform estimation in each group. However, when there are many attributes or attributes can take many unique values, the estimates may become inaccurate since there will be many groups and each group might only have a few nodes. In this case, we can make independence assumptions between attributes. For example, by assuming they are independent, the estimate for attribute combinations becomes the product of estimates for each attribute. The non-sensitive attributes can be safely removed under this assumption with $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv}) = 1$, and only $R(\boldsymbol{a}_{uv})$ needs to be estimated as $R(\boldsymbol{a}_{uv}) = \prod_{i=1}^{m} R(\boldsymbol{a}_{uv}[i])$. Since UGE-W only assigns pre-computed weights to the loss, the optimization based on it will not increase the complexity of any graph embedding method.

1.4.2 Regularization-Based UGE We propose an alternative way for UGE which adds a regularization term to the loss function that pushes the embeddings to satisfy properties required by the bias-free graph $\widetilde{\mathcal{G}}$. Specifically, when the node embeddings are learnt from $\widetilde{\mathcal{G}}$, their produced edge distribution should be the same with and without the sensitive attributes. To enforce this condition, we need to regularize the discrepancy between $P_o(E_{uv} = 1 | C_{uv} = \boldsymbol{a}_{uv}, \Theta_M)$ and $P_{\widetilde{o}}(E_{uv} = 1 | \widetilde{C}_{uv} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M)$ induced from the node embeddings. We can use the scores in $\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ as a proxy to represent edge probability between u and v, i.e., high $\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ indicates high probability of an edge between u and v. We can measure $P_o(E_{uv} = 1 | C_{uv} = \boldsymbol{a}_{uv}, \Theta_M)$ by aggregating all node pairs with the same attribute value combination to marginalize out the effect of Θ_M and focus on the influence from attribute value combinations as

$$
Q_{\boldsymbol{a}_{uv}} = \frac{1}{N_{\boldsymbol{a}_{uv}}} \sum_{i\in\mathcal{V}, j\in\mathcal{V}, \boldsymbol{a}_{ij} = \boldsymbol{a}_{uv}} \mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_i, \boldsymbol{z}_j), \tag{31}
$$

where we use $Q_{\boldsymbol{a}_{uv}}$ to denote the approximated measure of $P_o(E_{uv} = 1 | C_{uv} = \boldsymbol{a}_{uv}, \Theta_M)$, and $N_{\boldsymbol{a}_{uv}}$ is the number of node pairs that has the attribute value combination $\boldsymbol{a}_{uv}$. For pairs with the same attribute value combination, $Q_{\boldsymbol{a}_{uv}}$ only needs to be calculated once. Similarly, $P_{\widetilde{o}}(E_{uv} = 1 | \widetilde{C}_{uv} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M)$ can be represented by $Q_{\widetilde{\boldsymbol{a}}_{uv}}$, which can be obtained by aggregating the scores over pairs with non-sensitive attribute value combination $\widetilde{\boldsymbol{a}}_{uv}$. Finally, we use ℓ_2 distance between $Q_{\boldsymbol{a}_{uv}}$ and $Q_{\widetilde{\boldsymbol{a}}_{uv}}$ as the regularization

$$
\mathcal{L}_{UGE-R}(\mathcal{G}) = \sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) + \lambda \sum_{u\in\mathcal{V}, v\in\mathcal{V}} ||Q_{\boldsymbol{a}_{uv}} - Q_{\widetilde{\boldsymbol{a}}_{uv}}||_2, \tag{32}
$$

where λ controls the trade-off between the per-edge loss and the regularization.

Table 10: Statistics of evaluation graph datasets.

Statistics	Pokec-z	Pokec-n	MovieLens-1M
# of nodes	$67,796$	$66,569$	$9,992$
# of edges	$882,765$	$729,129$	$1,000,209$
Density	0.00019	0.00016	0.01002

In contrast to adversarial regularizations employed in prior work [112, 15, 4, 37], UGE-R takes a different perspective in regularizing the discrepancy between graphs with and without sensitive attributes induced from the embeddings. All previous regularization-based methods impose the constraint on individual edges. We should note the regularization term is summed over all node pairs, which has a complexity of $O(N^3)$ and can be costly to optimize. But in practice, we can add the regulariztaion by only sampling batches of node pairs in each iteration during model update, and use λ to compensate the strength of the regularization.

1.4.3 Combined Method As hinted in section 1, UGE-W is a sufficient condition for unbiased node embeddings and UGE-R is a necessary condition. We can combine them to trade-off the debiasing performance and utility,

$$\mathcal{L}_{UGE-C}(\mathcal{G}) = \sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})} + \lambda \sum_{u\in\mathcal{V}, v\in\mathcal{V}} ||Q_{\boldsymbol{a}_{uv}} - Q_{\widetilde{\boldsymbol{a}}_{uv}}||_2, \tag{33}$$

where we use $\mathcal{L}_{UGE-C}(\mathcal{G})$ to represent the combined method. $\mathcal{L}_{UGE-C}(\mathcal{G})$ thus can leverage the advantages of both UGE-W and UGE-R to achieve better trade-offs between the unbiasedness and the utility of node embeddings.

1.5 Experiments

In this section, we study the empirical performance of UGE on three benchmark datasets in comparison to several baselines. In particular, we apply UGE to five popularly adopted backbone graph embedding models to show its wide applicability. To evaluate the debiasing performance, the node embeddings are firstly evaluated by their ability to predict the value of sensitive attributes, where lower prediction performance means better debiasing effect. Then a task-specific metric is used to evaluate the utility of the embeddings. Besides, we also apply fairness metrics in the link prediction results to demonstrate the potential of using embeddings from UGE to achieve fairness in downstream tasks.

• **Datasets.** We used three public user-related graph datasets, Pokec-z, Pokec-n and MovieLens-1M, where the users are associated with sensitive attributes to be debiased. The statistics of these three datasets are summarized in Table 10. Pokec[7] is an online social network in Slovakia, which contains anonymized data of millions of users [162]. Based on the provinces where users belong to, we used two sampled datasets named as **Pokec-z** and **Pokec-n** adopted from [36], which consist of users belonging to two major regions of the corresponding provinces, respectively. In both datasets, each user has a rich set of features, such as education, working field, interest, etc.; and we include *gender, region* and *age* as (sensitive) attributes whose effect will be studied in our evaluation. **MovieLens-1M**[8] is a popular movie

[7] `https://snap.stanford.edu/data/soc-pokec.html`
[8] `https://grouplens.org/datasets/movielens/1m/`

Table 11: Unbiasedness evaluated by Micro-F1 on Pokec-z and Pokec-n. **Bold** highlights the best results.

Dataset	Prediction Target	No Debiasing	Fairwalk	CFC	UGE-W	UGE-R	UGE-C	Random
Pokec-z +GAT	Gender (Micro-F1)	0.6232	0.6135	0.5840	0.6150	0.6094	**0.5747**	0.4921
	Region (Micro-F1)	0.8197	0.8080	0.7217	0.6784	0.7660	**0.6356**	0.4966
	Age (Micro-F1)	0.0526	0.0522	0.0498	0.0431	0.0545	**0.0429**	0.0007
Pokec-n +node2vec	Gender (Micro-F1)	0.5241	0.5291	0.5241	0.5187	**0.5095**	0.5158	0.5078
	Region (Micro-F1)	0.8690	0.8526	0.8423	0.8158	0.6975	**0.6347**	0.4987
	Age (Micro-F1)	0.0626	0.0534	0.0426	0.0305	0.0294	**0.0194**	0.0002

recommendation benchmark, which contains around one million user ratings on movies [58]. In our experiment, we construct a bipartite graph which consists of user and movie nodes and rating relations as edges. The dataset includes *gender, occupation* and *age* information about users, which we treat as sensitive attributes to be studied. We do not consider movie attributes, and thus when applying UGE, only user attributes are counted for our debiasing purpose.

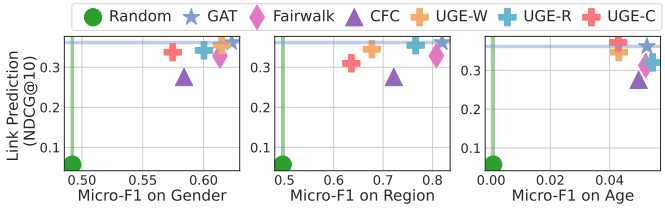
• **Graph Embedding Models.** UGE is a general recipe for learning unbiased node embeddings, and can be applied to different graph embedding models. We evaluate its effectiveness on five representative embedding models in the supervised setting with the link prediction task: **GCN** [81], **GAT** [174], **SGC** [190] and **node2vec** [52] are deep learning models, and we use dot product between two node embeddings to predict edge probability and apply cross-entropy loss for training. **MF** [123] applies matrix factorization to the adjacency matrix. Each node is represented by an embedding vector learnt with pairwise logistic loss [143].

• **Baselines.** We consider three baselines for generating unbiased node embeddings. (1) **Fairwalk** [138] is based on node2vec, which modifies the pre-processing of random-walk generation by grouping neighboring nodes with their values of the sensitive attributes. Instead of randomly jumping to a neighbor node, Fairwalk firstly jumps to a group and then sample a node from that group for generating random walks. We extend it to GCN, GAT and SGC by sampling random walks of size 1 to construct the corresponding per-edge losses for these embedding models. (2) **Compositional Fairness Constraints (CFC)** [15] is an algorithmic method, which adds an adversarial regularizer to the loss by jointly training a composition of sensitive attribute discriminators. We apply CFC to all graph embedding models and tune the weight on the regularizer in $\{1, 5, 10, 25, 35, 45, 55, 65\}$, where larger weights are expected to result in embeddings with less bias but also lower utility. (3) **Random** embeddings are used as a baseline which is considered bias-free. We generate random embeddings by sampling the value of each embedding dimension from [0, 1] uniformly at random.

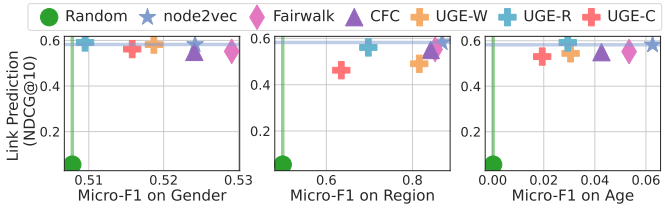
It is important to mention a recent work called **DeBayes** [19], which is based on the conditional network embedding (CNE) [77]. It includes the sensitive information in a biased prior for learning unbiased node embeddings. We did not include it since it is limited to CNE and cannot be easily generalized to other graph embedding models. Besides, we found the bias prior calculation in DeBayes does not scale to large graphs and the utility of resulting node embeddings is close to random. The original paper [19] only experimented with two small graph datasets with less than $4K$ nodes and $100K$ edges. By default, UGE follows Fairwalk to debias each of the three sensitive attributes separately without independence assumption between attributes. CFC debiases all sensitive attributes jointly as in the original paper[9].

• **Configurations.** For the Pokec-z and Pokec-n datasets, we apply GCN, GAT, SGC and node2vec as embedding models and apply debiasing methods on top of them. For each dataset, we construct positive

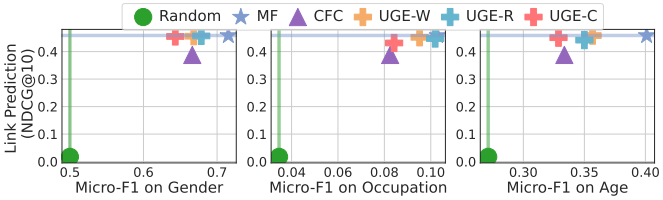
[9] UGE can also debias multiple attributes jointly.



(a) Pokec-z with GAT as embedding model



(b) Pokec-n with node2vec as embedding model



(c) MovieLens-1M with MF as embedding model

Fig. 16: Trade-off between utility by NDCG@10 (y-axis) and unbiasedness by micro-f1 (x-axis) of embeddings from different methods. Random embeddings without any bias give the lowest Micro-F1 (green line), and no debiasing gives the best NDCG@10 (blue line). An ideal debiasing method should locate itself at the upper left corner.

examples for each node by collecting N_{pos} neighboring nodes with N_{pos} equal to its node degree, and randomly sample $N_{neg} = 20 \times N_{pos}$ unconnected nodes as negative examples. For each node, we use 90% positive and negative examples for training and reserve the rest 10% for testing. For Movielens-1M, we follow common practices and use MF as the embedding model [138, 15] to evaluate CFC. We do not evalaute Fairwalk on this dataset since there is no user-user connections and fair random walk cannot be applied. The rating matrix is binarized to create a bipartite user-movie graph for MF. We use 80% ratings for training and 20% for testing. For all datasets and embedding models, we set the node embedding size to $d = 16$. Since there is a large number of experimental settings composed of different datasets, embedding models, and baselines, we report results from different combinations in each section to maximize the coverage in each component, and include the other results in Appendix 2.

1.5.1 Analysis of Unbiasedness We first compare the unbiasedness of node embeddings from different debiasing methods. For each sensitive attribute, we train a logistic regression classifier with randomly sampled 80% nodes with their embeddings as features and attribute values as class labels. Then we use the classifier to predict the attribute values of the rest of 20% nodes and evaluate the performance with Micro-F1. The Micro-F1 score can be used to measure the bias in the embeddings, i.e., a lower score means lower bias in the embedding with respect to the sensitive attribute. In expectation, random embeddings should have the lowest Micro-F1 and embeddings without debiasing should have the high-

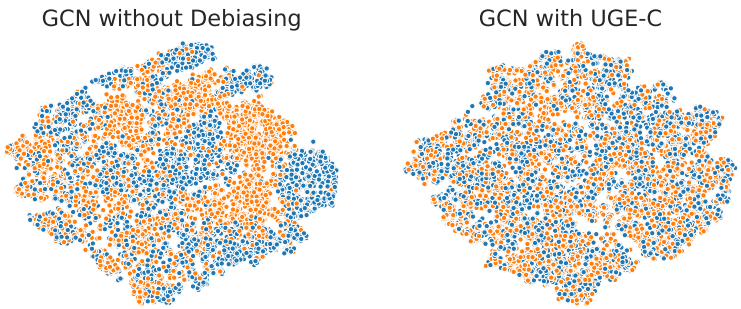


Fig. 17: Visualizing embeddings learnt on Pokec-n using GCN. Node color represents region attribute.

est Micro-F1. We show the results on Pokec-z with GAT as base embedding model and Pokec-n with node2vec as base embedding model in Table 11.

From the results, we can find that embeddings from UGE methods always have the least bias against all baselines with respect to all sensitive attributes and datasets. This confirms the validity of learning unbiased embeddings from a bias-free graph regarding the sensitive attributes. Besides, by combining weighting and regularization, UGE-C usually produces the best debiasing effect, which demonstrates the complementary effect of our two methods.

Besides the unbiasedness, the learnt embeddings are required to be effective when applied to downstream tasks. In particular, we use NDCG@10 evaluated on the link prediction task to measure the utility of the embeddings. Specifically, for each target node, we create a candidate list of size 100 that includes all its observed neighbor nodes in the test set and randomly sampled negative nodes. Then NDCG@10 is evaluated on this list with predicted edge probabilities from node embeddings. Figures 16(a) and 16(b) show the unbiasedness as well as the utility of embeddings from different methods in correspondence to the two datasets and embedding models in Table 11. Figure 16(c) shows the results on MovieLens-1M with MF as embedding model.

In these plots, the y-axis denotes NDCG@10 evaluating the utility of embeddings and x-axis shows Micro-F1 measuring the bias in embeddings. Different embedding methods are represented by different shapes in the figures, and we use different colors to differentiate UGE-W, UGE-R and UGE-C. Again, random embeddings do not have any bias and provide the lowest Micro-F1 (denoted by the green line), while embeddings without any debiasing gives the best NDCG@10 (denoted by the blue line). An ideal debiasing method should locate at the upper left corner to achieve the best utility-unbiasedness trade-off. As shown in the figures, UGE methods achieve the most encouraging trade-off on these two contradicting objectives in most cases. UGE-C can usually achieve better debiasing effect, without sacrificing too much utility in most cases. UGE-W and UGE-R maintain high utility but are less effective than the combined version. CFC can achieve descent unbiasedness in embeddings, but the utility is seriously compromised (such as in Pokec-z and MovieLens-1M). Fairwalk unfortunately does not present an obvious debiasing effect on sensitive attributes.

To further illustrate the debiasing effect from UGE, we use the t-SNE algorithm to project the learned node embeddings on Pokec-n to a 2-D space and visualize them in Figure 17. The left plot shows the embedding learned via GCN without debiasing, and the right plot exhibits the debiased embeddings by applying UGE-C on GCN to debias the region attribute. Node colors represent the region value. Without debiasing, the embeddings are clearly clustered with respect to region. After applying UGE-C, embeddings with different region values are blended together, showing the effect of removing the region information from embeddings.

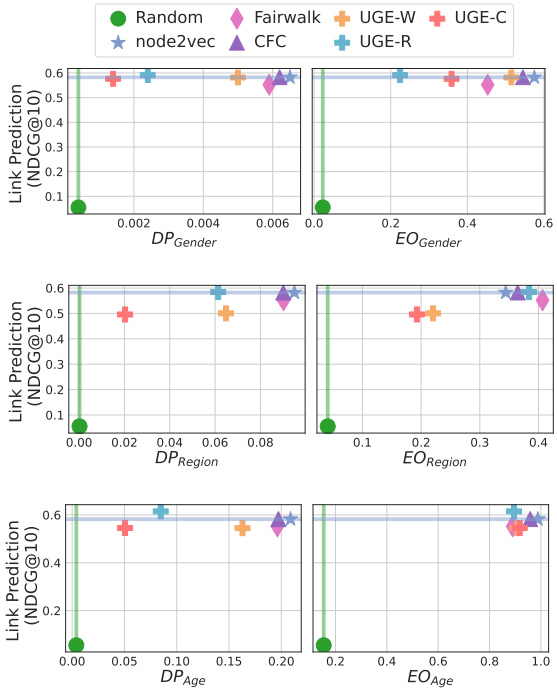


Fig. 18: Fairness metrics evaluated on link prediction task on Pokec-n with node2vec embedding model.

1.5.2 High-Level Fairness from Embeddings We study whether the embeddings can lead to fairness in downstream tasks. Two fairness metrics called *demographic parity* (DP) and *equalized opportunity* (EO) are adopted to evaluate the fairness in the link prediction task from the embeddings. In particular, DP requires that the predictions are independent from sensitive attributes, measured by the maximum difference of prediction rates between different combinations of the sensitive attribute values. EO measures the independence between true positive rate (TPR) of predicting edges and the sensitive attributes, and it is defined as the maximum difference of TPRs between different sensitive attribute value combinations. For both DP and EO, lower values suggest better fairness. We use the exact formulation of DP and EO in [19] and use sigmoid function to convert each edge score for a pair of nodes to a probability.

We show the results on fairness vs., utility in Figure 18, which are evaluated on three sensitive attributes in Pokec-n with node2vec as embedding model. In each plot, the x-axis is the fairness metric and y-axis is the NDCG@10 on link prediction. Similar to Figure 16, the ideal debiasing methods should locate at the upper left corner. Except for EO on the *age* attribute where all methods performed similarly, UGE methods can achieve significantly better fairness than the baselines on both DP and EO, while maintaining competitive performance on the link prediction task. UGE-C that combines UGE-W and UGE-R can achieve the most fair predictions. This study shows UGE's ability to achieve fairness in downstream tasks by effectively eliminating bias in the learnt node embeddings.

1.5.3 Unbiasedness-Utility Tradeoff in UGE Last but not least, we study the unbiasedness-utility trade-off in UGE-C by tuning the weight on the regularization. Although UGE-W itself can already achieve promising debiasing effect, we expect that the added regularization UGE-R can complement it for a better trade off. In particular, we tuned the regularization weights λ in both CFC and UGE-C

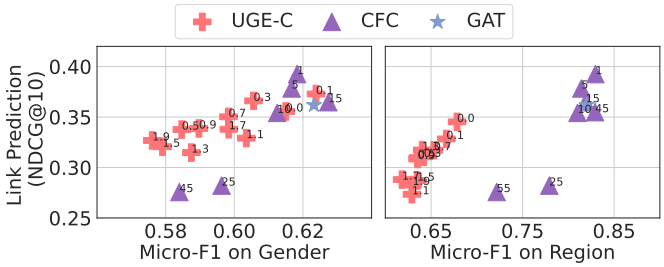


Fig. 19: Trade-off comparison between CFC and UGE-C on Pokec-z with GAT as the embedding model.

and plot Micro-F1 (x-axis) vs., NDCG@10 (y-axis) from the resulting embeddings in Figure 19. Weight values are marked on each point. The results are obtained on Pokec-z with GAT and the two figures correspond to debiasing *gender* and *region*, respectively. With the same extent of bias evaluated by Micro-F1, embeddings from UGE-C have a much higher utility as indicated by the vertical grids. On the other hand, embeddings from UGE-C have much less bias when the utility is the same as indicated by horizontal grids. This experiment proves a better trade-off is achieved in UGE-C, which is consistent with our design on UGE-W and UGE-R. UGE-W learns from a bias-free graph without any constraints, and it is sufficient for it to achieve unbiasedness without hurting the utility of embeddings. On the other hand, UGE-R requires the embeddings to satisfy the properties of those learnt from the bias-free graph, which is necessary for the embeddings to be unbiased.

2 Robust Machine Learning with Perturbed Structure

Graph data differs from image or text data due to the topological structure formed among nodes. On one hand, GCNs exploit such structures to aggregate information conveyed in nodes' neighborhoods, which yield better predictive power on many tasks (such as link prediction [151] and node classification [145]). But on the other hand, the complex dependency relations introduced by the topological structure of graphs also expose learning models to a greater risk: an attacker can mislead classifiers to erroneous predictions by just slightly perturbing the graph structure, without even modifying any node features. Various adversarial attacks have been studied on graph structure [72], considering different prediction tasks (node classification [193, 180, 209] or graph classification [38, 111]), attacker's knowledge (white-box [193, 180, 208] or black-box [38, 111]), attack phases (test-time evasion attack [24, 38, 193] or training-time poisoning attack [180, 209]), and perturbation types (edge modification [193, 209] or node modification [160]). In this paper, we focus on the structural attack by adding or removing edges to compromise the node classification performance of a victim GCN model.

Graph convolution, as the fundamental building block of GCNs, is designed to filter graph signals in the *Fourier* domain. Studies in spectral graph theory [33] show that the spectra (eigenvalues) of the graph Laplacian matrix capture graph structural properties (e.g., the second smallest eigenvalue, also known as the Fiedler value, reflects the algebraic connectivity of graphs [124]). Therefore exploiting spectral changes provides a comprehensive way to study the vulnerability of GCN models. However, so far most structural attack solutions only search for perturbations in the *spatial* domain. Ignoring the direct source of GCN models' vulnerability which resides in the Fourier domain limits the effectiveness of attacks.

Studying GCN models' vulnerability in the Fourier domain can effectively capture important edges that influence the structural property the most, e.g., the clustering structure of nodes. According to the concept of graph signal processing, the eigen-decomposition of the Laplacian matrix of a graph defines the

frequency domain of message passing on the graph. Recent works have established the relationship between frequency components and graph clustering [40, 155]. Based on the ascending ordered eigenvalues of the Laplacian matrix, we can obtain both low- and high-frequency components, which play different roles in message passing on graphs. The eigenvectors associated with small eigenvalues carry smoothly varying signals, encouraging neighbor nodes to share similar properties (e.g., nodes within a cluster). In contrast, the eigenvectors associated with large eigenvalues carry sharply varying signals across edges (e.g., nodes from different clusters) [23, 70]. The fact that graph frequency components encode the global structure of graphs motivates us to study GCN models' vulnerability in the Fourier domain.

In this work, we propose a principled graph perturbation strategy in the Fourier domain to improve the effectiveness of adversarial attacks against GCN models. Specifically, we define the *spectral distance* between the original and perturbed graph, measured by the change in their Laplacian eigenvalues. Then we build a structural attack model which directly maximizes the spectral distance in a black-box fashion. To solve this combinatorial optimization problem, we relax the binary constraint on edge perturbation to a continuous one, and apply a randomization sampling strategy to generate valid binary edge perturbations. We name this method **SP**ectral **A**tta**C**k, abbreviated as **SPAC**. It is worth noting that generating the SPAC attack requires eigen-decomposition of the Laplacian matrix, which results in a time complexity of $\mathcal{O}(n^3)$ with n nodes in a graph. To handle large graphs, we propose an approximation solution only based on a set of largest and smallest eigenvalues and their corresponding eigenvectors, and use eigenvalue perturbation theory [157] to avoid frequent computation of eigen-decomposition, which reduces the time complexity to $\mathcal{O}(n^2)$. Our attack method is evaluated under both white-box and black-box settings for both evasion and poisoning attacks on a set of benchmark graph datasets. Promising empirical results demonstrate that convolutional graph learning models are sensitive to spectral changes, which expands the scope of adversarial attacks on graphs to the Fourier domain and opens up new possibilities to verify and enhance GCNs' robustness in both the spatial and Fourier domains.

2.1 Related Work

Adversarial attacks on graph structures have been extensively studied in recent years. The vast majority of attack efforts manipulate graphs in the spatial domain to maximize a task-specific attack objective. However, the vulnerability of graph convolutions in the Fourier domain is less studied in existing attack solutions. We bridge the gap by measuring and maximizing the spectral changes in the graph Laplacian matrix, such that we can directly disrupt the graph spectral filters and attack the graph convolutions.

• **Adversarial Attack on Graph Structures.** The attacker aims to perturb the graph adjacency matrix in directions that lead to large classification loss. In the white-box setting, the attacker follows the gradients on the adjacency matrix to find such perturbations [209, 193, 191, 208, 28, 95, 181]. Different strategies are exploited to convert continuous gradients into binary edge modifications. Topology attack [113] uses randomization sampling to select sub-optimal binary perturbations. Nettack [208] and FGA [28] select edge changes with the largest gradient greedily. Metattack [209] first calculates meta-gradient on graph adjacency matrix to solve a bi-level optimization problem for poisoning attack, and then greedily picks perturbations with the largest meta-gradient. In the black-box setting, the attacker cannot access gradients of the victim model but uses a proxy (e.g. model output scores) to search for the best perturbations [109, 111, 38]. Reinforcement learning based solutions [24, 111, 38, 139] make a series of edge addition or deletion decisions that yield the maximal attack utility and thus can serve for black-box setting.

These attacks search for perturbations in the spatial space, but the target GCNs generate node embeddings by the signal filters defined in the Fourier space. Thus the vulnerability of graph convolutions reflected on the graph spectral changes cannot be fully realized. Our method captures such vulnerability directly

in the Fourier domain measured by the spectral distance between the original and perturbed graphs for a more effective attack.

• **Spectral Perturbations on Graphs.** Existing attack methods in the Fourier space are generally sparse. Bojchevski and Günnemann [14] reformulate random walk based models as a matrix factorization problem, and propose an attack strategy to search for edges that lead to large eigenvalue changes in the derived matrix. However, this method is model-specific and cannot be easily applied to general forms of GCNs. GF-Attack [24] constructs an objective based on the low-rank graph spectra and feature matrix to guide the attack in a black-box fashion. A universal attack on deformable 3D shape data is proposed to change the scale of its eigenvalues [140], but it is not studied in the graph domain. DICE [186] corrupts the graph structure by "deleting edges internally and connecting nodes externally" across clusters which implicitly influences the graph's spectral property. But this heuristic is performed without any principled guidance. Studies that analyze spectral graph filters [79, 92, 78] provide the theoretical stability upper bounds of popular filters, such as polynomial and identity filters. It is shown that the filters become unstable if the end nodes of changed edges have low degrees or the perturbation is concentrated spatially around any single node [79]. Our method empirically shows that we can attack the vulnerability of these filters and break such requirements by directly maximizing graph spectral changes.

2.2 Preliminaries: Graph Neural Networks from Signal Processing Perspective

• **Notations.** Let $G = (V, E)$ be a connected undirected graph with n nodes and m edges. Let $\mathbf{A} \in \{0, 1\}^{n\times n}$ be its adjacency matrix. The diagonal degree matrix can be calculated by $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_n)$ with entry $\mathbf{D}_{ii} = d_i = \sum_{j=1}^{n} \mathbf{A}_{ij}$, and $\mathbf{1}_n$ is an all-one vector with dimension n. The normalized Laplacian matrix of a graph is defined as $\mathbf{L} = \text{Lap}(\mathbf{A}) = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix. Since $\mathbf{L}$ is symmetric positive semi-definite, it admits an eigen-decomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. The diagonal matrix $\mathbf{\Lambda} = \text{eig}(\mathbf{L}) = \text{diag}(\lambda_1, \ldots, \lambda_n)$ consists of the real eigenvalues of $\mathbf{L}$ in an increasing order such that $0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$, and the corresponding $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n\times n}$ is a unitary matrix where the columns consist of the eigenvectors of $\mathbf{L}$. $\mathbf{X} \in \mathbf{R}^{n\times d}$ denotes the node feature matrix where each node v is associated with a d-dimensional feature vector.

• **Graph Fourier Transform.** By viewing graph embedding models from a signal processing perspective, the normalized Laplacian $\mathbf{L}$ serves as a shift operator and defines the frequency domain of a graph [149]. As a result, the eigenvectors of $\mathbf{L}$ can be considered as the graph Fourier bases, and the eigenvalues correspond to frequency components. Take one column of $\mathbf{X}$ as an example of graph signal, which can be compactly represented as $\mathbf{x} \in \mathbb{R}^n$. The graph Fourier transform of graph signal $\mathbf{x}$ is given by $\widehat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$ and the inverse graph Fourier transform is then $\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}}$. The graph signals in the Fourier domain are filtered by amplifying or attenuating the frequency components $\mathbf{\Lambda}$.

• **Spectral Graph Convolution.** At the essence of different graph convolutional models is the spectral convolution, which is defined as the multiplication of signal $\mathbf{x}$ with a filter g_ϕ parameterized by $\phi \in \mathbb{R}^n$ in the Fourier domain [39]:

$$g_\phi(\mathbf{L}) \star \mathbf{x} = \mathbf{U}g^*_\phi(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{x} \tag{34}$$

where the parameter ϕ is a vector of spectral filter coefficients. The filter g_ϕ defines a smooth transformation function, and a commonly used filter is the polynomial filter:

$$g^*_\phi(\mathbf{\Lambda}) = \sum\nolimits_{k=0}^{\infty} \phi_k\mathbf{\Lambda}^k \tag{35}$$

which can be approximated by a truncated expansion. A commonly adopted approximation is based on the Chebyshev polynomials $T_k(\mathbf{\Lambda})$, which are recursively defined as $T_0(\mathbf{\Lambda}) = \mathbf{I}_n, T_1(\mathbf{\Lambda}) = \mathbf{\Lambda}$ and

$T_{k+1}(\mathbf{\Lambda}) = 2\mathbf{\Lambda}T_k(\mathbf{\Lambda}) - T_{k-1}(\mathbf{\Lambda})$. Using the Chebyshev polynomials up to the K-th order achieves the following approximation [56]:

$$g^*_\phi(\mathbf{\Lambda}) \approx \sum\nolimits_{k=0}^{K} \phi_k T_k(\widetilde{\mathbf{\Lambda}}) \tag{36}$$

with a rescaled $\widetilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_n - \mathbf{I}_n$.

Graph Convolutional Network (GCN). A vanilla GCN is a first-order approximation of the spectral graph convolution with the Chebyshev polynomials [81]. Setting $K = 1$, $\phi_0 = -\phi_1$ in Eq (36) and approximating $\lambda_n \approx 2$, we obtain the convolution operation $g_\phi(\mathbf{L}) \star \mathbf{x} = (\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$. We can replace matrix $\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ with a self-loop version $\widetilde{\mathbf{L}} = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}$ where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ and $\widetilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}_n$. This resembles the vanilla GCN layer with activation function σ and trainable network parameters $\mathbf{\Theta}$ for feature transformation: $\mathbf{H}^{(l+1)} = \sigma\Big(\widetilde{\mathbf{L}}\mathbf{H}^{(l)}\mathbf{\Theta}^{(l)}\Big)$, where the signals from the previous layer $\mathbf{H}^{(l)}$ is filtered to generate new signals $\mathbf{H}^{(l+1)}$. To unify the notations, $\mathbf{H}^{(0)} = \mathbf{X}$ denotes the input node features, and $\mathbf{Z} = \mathbf{H}^{(L)}$ denotes the output node embeddings of an L-layer GCN model.

2.3 Methodology

We propose to maximize the changes on the graph Laplacian spectrum, such that we can exploit the edge perturbation budget to most effectively influence the spectral filters and attack graph convolutions. We solve the resulting optimization problem using gradient descent, and propose an efficient approximation via eigenvalue perturbation theory on selective eigenvalues. We finally discuss the theoretical evidence showing the dependency between the eigenvalues of graph Laplacian and the stability of GCN models, which supports the proposed spectral attack on graph data. Based on the aforementioned spectral perspective for understanding GCNs, we aim to generate edge perturbations that can disrupt the spectral filters the most when processing input signals on the graph. We measure the disruption by the changes in the eigenvalues of graph Laplacian, which we define as the spectral distance.

2.3.1 Spectral Distance on Graphs As shown in Eq (34), the spectral filters $g^*_\phi(\mathbf{\Lambda})$ are the key in graph convolutions to encode graph signals that are transformed in the Fourier domain. The output of the spectral filters is then transformed back to the spatial domain to generate node embeddings. Therefore, perturbing the spectral filters $g^*_\phi(\mathbf{\Lambda})$ will affect the filtered graph signals and produce inaccurate node embeddings. To measure the changes in spectral filters, we define the spectral distance between the original graph G and perturbed graph G' as:

$$\mathcal{D}_{\text{spectral}} = \|g^*_\phi(\mathbf{\Lambda}) - g^*_\phi(\mathbf{\Lambda}')\|_2 \tag{37}$$

where $\mathbf{\Lambda}$ and $\mathbf{\Lambda}'$ are the eigenvalues of the normalized graph Laplacian for G and G' respectively. The spectral distance $\mathcal{D}_{\text{spectral}}$ is determined by both filter parameters ϕ and the frequency components $\mathbf{\Lambda}$. For graph embedding models based on the vanilla GCN [81], we follow their design of spectral filters which uses the first-order approximation of the Chebyshev polynomials in Eq (36) and sets $\phi_0 = -\phi_1 = 1$, which gives:

$$g^*_\phi(\mathbf{\Lambda}) \approx \phi_0\mathbf{I}_0 + \phi_1\mathbf{\Lambda} = \mathbf{I}_n - \mathbf{\Lambda} \tag{38}$$

Plugging it into Eq (37), we conclude the following spectral distance which is only related to the eigenvalues of graph Laplacian:

$$\mathcal{D}_{\text{spectral}} \approx \|(\mathbf{I}_n - \mathbf{\Lambda}) - (\mathbf{I}_n - \mathbf{\Lambda}')\|_2 = \|\mathbf{\Lambda} - \mathbf{\Lambda}'\|_2 = \|\text{eig}(\text{Lap}(\mathbf{A})) - \text{eig}(\text{Lap}(\mathbf{A}'))\|_2 \tag{39}$$

This spectral distance reflects the changes of spectral filters due to the graph perturbation. Therefore, if we perturb the graph by directly maximizing the spectral distance, we can impose the most effective changes to graph filters and thus disrupt the generated node embeddings the most.

2.3.2 Spectral Attack on Graph Structure Since the spectral distance measures the changes of the spectral filters on the graph after perturbation, we can produce effective attacks by maximizing the resulting spectral distance. The goal is to find the perturbation on the graph adjacency matrix that can maximize the spectral distance $\mathcal{D}_{\text{spectral}}$ defined in Eq (39). We first define the *structural perturbation* as a binary perturbation matrix $\mathbf{B} \in \{0,1\}^{n\times n}$, which indicates where to flip the edges in G. The new adjacency matrix after the perturbation is then a function of the perturbation matrix, which can be obtained as follows [193]:

$$g(\mathbf{A}, \mathbf{B}) = \mathbf{A} + \mathbf{C} \circ \mathbf{B},\ \mathbf{C} = \bar{\mathbf{A}} - \mathbf{A} \tag{40}$$

where $\bar{\mathbf{A}}$ is the complement matrix of the adjacency matrix $\mathbf{A}$, calculated by $\bar{\mathbf{A}} = \mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n - \mathbf{A}$, with $(\mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n)$ denoting the fully-connected graph without self loops. Therefore, $\mathbf{C} = \bar{\mathbf{A}} - \mathbf{A}$ denotes legitimate addition or deletion operations on each node pair: adding an edge is allowed between node i and j if $\mathbf{C}_{ij} = 1$, and removing an edge is allowed if $\mathbf{C}_{ij} = -1$. Taking the Hadamard product $\mathbf{C} \circ \mathbf{B}$ finally gives valid edge perturbations to the graph.

To generate an effective structural attack, we seek a perturbation matrix $\mathbf{B}$ that maximizes the spectral distance $\mathcal{D}_{\text{spectral}}$ defined in Eq (39). More specifically, given a finite budget of edge perturbation, e.g., $\|\mathbf{B}\|_0 \leq \epsilon|E|$ with $|E|$ denoting the number of edges, we formulate the **SP**ectral **A**tta**C**k (**SPAC**) as the following optimization problem:

$$\max_{\mathbf{B}} \mathcal{L}_{\text{SPAC}} \coloneqq \mathcal{D}_{\text{spectral}} \quad \text{s.t. } \|\mathbf{B}\|_0 \leq \epsilon|E|, \mathbf{B} \in \{0,1\}^{n\times n}, \mathbf{A}' = g(\mathbf{A}, \mathbf{B}) \tag{41}$$

which is not straightforward to solve because of two challenges: 1) it is a combinatorial optimization problem due to the binary constraint on $\mathbf{B}$; 2) the objective involves eigen-decomposition of the Laplacian matrix, which is time-consuming especially for large graphs. Next, we introduce our practical solution to address these challenges such that we can efficiently generate the structural perturbations for the spectral attack.

2.4 Practical implementation of SPAC

To solve the combinatorial optimization problem involving eigen-decomposition in Eq (41), we first relax the combinatorial problem and use a randomization sampling strategy to generate the binary perturbation matrix; we then introduce an approximation strategy to reduce the complexity of backpropagation through eigen-decompostion.

2.4.1 Binary perturbation via gradient descent For the ease of optimization, we relax $\mathbf{B} \in \{0,1\}^{n\times n}$ to its convex hull $\mathbf{\Delta} \in [0,1]^{n\times n}$ [193], which simplifies the combinatorial problem in Eq (41) to be the following continuous optimization problem:

$$\max_{\mathbf{\Delta}} \mathcal{L}_{\text{SPAC}} \coloneqq \mathcal{D}_{\text{spectral}} \quad \text{s.t. } \|\mathbf{\Delta}\|_1 \leq \epsilon|E|, \mathbf{\Delta} \in [0,1]^{n\times n}, \mathbf{A}' = g(\mathbf{A}, \mathbf{\Delta}) \tag{42}$$

which can be solved via gradient descent. The gradient with respect to $\mathbf{\Delta}$ is as follows by chain rule:

$$\frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \mathbf{\Delta}_{ij}} = \sum_{k=1}^{n} \frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \lambda'_k} \sum_{p=1}^{n}\sum_{q=1}^{n} \frac{\partial \lambda'_k}{\partial \mathbf{L}'_{pq}} \frac{\partial \mathbf{L}'_{pq}}{\partial \mathbf{\Delta}_{ij}} \tag{43}$$

Recall that $\mathbf{L}' = \text{Laplacian}(\mathbf{A}')$ and λ'_k is an eigenvalue of $\mathbf{L}'$ in Eq (39). We now focus on the gradient calculation that involves eigen-decomposition. Since the gradient calculation of the rest is straightforward, we leave it to Appendix 3.1. For a real and symmetric matrix $\mathbf{L}$, one can obtain the derivatives of its eigenvalue λ_k and eigenvector $\mathbf{u}_k$ by: $\partial\lambda_k/\partial\mathbf{L} = \mathbf{u}_k\mathbf{u}_k^\top, \partial\mathbf{u}_k/\partial\mathbf{L} = (\lambda_k\mathbf{I} - \mathbf{L})^+\mathbf{u}_k$ [144, 114]. Therefore, we can directly obtain the closed-form derivative of the eigenvalues in Eq (43) as: $\partial\lambda'_k/\partial\mathbf{L}'_{pq} = \mathbf{u}'_{kp}\mathbf{u}'_{kq}$. Note that the derivative calculation requires distinct eigenvalues. This does not hold for graphs satisfying *automorphism*, which reflects structural symmetry of graphs [47]. To avoid such cases, we add a small noise term to the adjacency matrix of the perturbed graph[10], e.g., $\mathbf{A}' + \varepsilon \times (\mathbf{N} + \mathbf{N}^\top)/2$, where each entry in $\mathbf{N}$ is sampled from a uniform distribution $\mathcal{U}(0, 1)$ and ε is a very small constant. Such a noise addition will almost surely break the graph automorphism, thus enable a valid gradient calculation of eigenvalues.

Solving the relaxed problem in Eq (42) using projected gradient descent gives us a continuous perturbation matrix $\mathbf{\Delta}$ that maximizes the spectral change. To recover valid edge perturbations from the continuous $\mathbf{\Delta}$, we then generate a near-optimal solution for the binary perturbation matrix $\mathbf{B}$ via the randomization sampling strategy [193]. Specifically, we use $\mathbf{\Delta}$ as a probabilistic matrix to sample the binary assignments as follows:

$$\mathbf{B}_{ij} = \begin{cases} 1, & \text{with probability } \mathbf{\Delta}_{ij} \\ 0, & \text{with probability } 1 - \mathbf{\Delta}_{ij} \end{cases} \tag{44}$$

Suppose we take aforementioned projected gradient descent for T steps. For each step, SPAC takes eigen-decomposition with time complexity $\mathcal{O}(n^3)$ and samples binary solution with $\mathcal{O}(n^2)$ edge flips. The overall time complexity for solving SPAC is $\mathcal{O}(Tn^3 + Tn^2)$, which is mainly attributed to eigen-decomposition and is considerably expensive for large graphs. Next, we discuss an approximation solution to improve its efficiency.

2.4.2 Efficient approximation for SPAC To reduce the computation cost of eigen-decomposition, instead of measuring the spectral distance over all the frequency components, we decide to only maintain the k_1 lowest- and k_2 highest-frequency components which are the most informative, as suggested by the spectral graph theory. Specifically, $\mathcal{D}_{\text{spectral}}$ in Eq (39) can be approximated as follows:

$$\mathcal{D}_{\text{spectral-approx}} = \sqrt{\sum_{i \in \mathcal{S}} (\lambda_i - \lambda'_i)^2} \tag{45}$$

where $\mathcal{S} = \{1, \ldots, k_1, n - k_2, \ldots, n\}$. This reduces the time complexity $\mathcal{O}(n^3)$ for exact eigen-decomposition to $\mathcal{O}((k_1 + k_2) \cdot n^2)$ for the corresponding selective eigen-decomposition using the Lanczos Algorithm [131]. To avoid frequent computation of the selective eigenvalues and further improve efficiency, we propose to estimate the change in each eigenvalue for any edge perturbation based on the eigenvalue perturbation theory [141, 14, 24].

Theorem 1. *Let $\mathbf{u}_i$ be the i-th generalized eigenvector of $\mathbf{L}$ with generalized eigenvalue λ_i. Let $\mathbf{L}' = \mathbf{L} + \nabla\mathbf{L}$ be the perturbed Laplacian matrix, and $\mathbf{M}'$ be the diagonal matrix summing over rows of $\mathbf{L}'$. The generalized eigenvalue λ'_i of the Laplacian matrix $\mathbf{L}'$ that solves $\mathbf{L}'\mathbf{u}'_i = \lambda'_i\mathbf{M}'\mathbf{u}'_i$ is approximately $\lambda'_i \approx \lambda_i + \nabla\lambda_i$ with:*

$$\lambda_i - \lambda'_i = \nabla\lambda_i = \mathbf{u}_i^\top(\nabla\mathbf{L} - \lambda_i \mathit{diag}(\nabla\mathbf{L} \cdot \mathbf{1}_n))\mathbf{u}_i \tag{46}$$

[10] The form of $(\mathbf{N} + \mathbf{N}^\top)/2$ is to keep the perturbed adjacency matrix symmetric for undirected graphs.

Algorithm 1 Spectral Attack on Graph Structure

Input: $G = (\mathbf{X}, \mathbf{A})$; total step T; step size η; k_1, k_2, m.

1: Initialize continuous perturbation $\mathbf{\Delta}_0 \in [0,1]^{n\times n}$
2: Initialize perturbed Laplacian matrix $\mathbf{L}' = \text{Laplacian}(g(\mathbf{A}, \mathbf{\Delta}))$
3: **for** $t = 0, \ldots, T-1$ **do**
4: **if** SPAC **then**
5: $\mathcal{L}(\mathbf{\Delta}) = \|\mathbf{\Lambda} - \mathbf{\Lambda}'\|_2$ by Eq (42), with $\mathbf{\Lambda}' = \text{eig}(\mathbf{L}')$
6: **else if** SPAC-approx **then**
7: **if** $t \,\%\, m = 0$ **then**
8: $\mathcal{L}(\mathbf{\Delta}) = \sqrt{\sum_{i\in\mathcal{S}}(\lambda_i - \lambda_i')^2}$ by Eq (45)
9: **else**
10: $\mathcal{L}(\mathbf{\Delta}) = \sqrt{\sum_{i\in\mathcal{S}}(\mathbf{u}_i^\top(\nabla\mathbf{L} - \lambda_i \text{diag}(\nabla\mathbf{L}\cdot\mathbf{1}_n))\mathbf{u}_i)^2}$ by Eq (46) and Eq (45), with $\nabla\mathbf{L} = \mathbf{L}' - \mathbf{L}$
11: **end if**
12: **end if**
13: Compute gradient on $\mathbf{\Delta}$: $\mathbf{g}_t = \nabla\mathcal{L}(\mathbf{\Delta})$ by Eq (43)
14: Update $\mathbf{\Delta}_{t+1} = \mathbf{\Delta}_t + \eta \cdot \mathbf{g}_t$
15: Project $\mathbf{\Delta}_{t+1}$ to its convex hull $\mathbf{\Delta}_{t+1} \in [0,1]^{n\times n}$
16: **end for**
17: Output binary perturbation $\mathbf{B}$ by sampling from $\mathbf{\Delta}_T$ via Eq (44)

The proof is given in Appendix 3.2. Instead of recalculating the eigenvalues λ_i' for the updated $\mathbf{L}'$ in each step when measuring the spectral distance, we use this theorem to approximate the change of each eigenvalue in Eq (45) in linear time. Suppose we execute Eq (46) to calculate spectral distance in Eq (45) for m steps and compute the exact eigenvalues every m step to avoid error accumulation, we can achieve an overall time complexity $\mathcal{O}\big((1 + \frac{k_1+k_2}{m})Tn^2\big)$. We name the spectral attack equipped with the approximation stated in Eq (45) and Eq (46) as **SPAC-approx**.

Algorithm 1 summarizes the implementation of SPAC (in line 5) and its approximation SPAC-approx (in line 7-10). After obtaining the objective function based on the (approximated) spectral distance, the algorithm further updates the continuous perturbation matrix $\mathbf{\Delta}$ via gradient descent and finally generates the binary structural perturbation matrix $\mathbf{B}$ by sampling from $\mathbf{\Delta}$.

2.4.3 Extension in White-box Setting The proposed attack only requires information about graph spectrum, therefore it can be conducted alone in the black-box setting as Eq (41) stated. Since SPAC does not rely on any specific embedding model, it can also serve as a general recipe for the white-box attack setting. Next, we show how to easily combine SPAC with white-box attack models.

Without loss of generality, we consider the vanilla GCN model for the node classification task as the *Victim Graph Embedding Model*. Given a set of labeled nodes $V_0 \subset V$, where each node i belongs to a class in a label set $y_i \in Y$. The GCN model aims to learn a function f_θ that maps each node to a class. We consider the commonly studied transductive learning setting, where the test (unlabeled) nodes with associated features and edges are observed in the training phase. The GCN model is trained by minimizing the following loss function:

$$\min_\theta \ \mathcal{L}_{\text{train}}(f_\theta(G)) = \sum_{v_i\in V_0} \ell(f_\theta(\mathbf{A}, \mathbf{X})_i, y_i)$$

where $f_\theta(\mathbf{X}, \mathbf{A})_i$ and y_i are the predicted and ground-truth labels of node v_i and $\ell(\cdot, \cdot)$ is a loss function of choice, such as the cross entropy loss.

To generate edge perturbations under *white-box setting* that lead to both disrupted spectral filters and erroneous classifications, we propose to maximize the spectral distance and task-specific attack objective simultaneously. Specifically, given the test node-set $V_t \subset V$, the attack model aims to find the edge perturbation $\mathbf{B}$ that solves the following optimization problem:

$$\begin{aligned} \max_{\mathbf{B}} \; & \underbrace{\sum_{v_i \in \mathcal{V}_t} \ell_{\text{atk}}(f_{\theta^*}(\mathbf{A}', \mathbf{X})_i, y_i)}_{\text{task-specific attack objective } \mathcal{L}_{\text{attack}}} + \beta \cdot \mathcal{L}_{\text{SPAC}} \\ \text{s.t.} \; & \|\mathbf{B}\|_0 \leq \epsilon, \mathbf{B} \in \{0,1\}^{n \times n}, \mathbf{A}' = g(\mathbf{A}, \mathbf{B}) \\ & \theta^* = \arg\min_{\theta} \; \mathcal{L}_{\text{train}}(f_\theta(\widehat{G})) \end{aligned} \tag{47}$$

where the third constraint controls when to apply the attack: setting $\widehat{G} = G$ makes it an evasion attack, so that the graph embedding model training is not affected; and setting $\widehat{G} = G'$ makes it poisoning attack with perturbed training data. The attack objective $\mathcal{L}_{\text{attack}}$ is a flexible placeholder that can adapt to many loss designs, for a simple example, the cross-entropy loss on test nodes in the node classification task. The hyper-parameter β balances the effect of these two components, which is set based on graph properties such as edge density. Algorithm 1 also applies for the white-box setting by plugging in $\mathcal{L}_{\text{attack}}$ to the objective function. We will discuss the choices of attack objectives and hyper-parameters in the experiment section for our empirical evaluations.

2.5 Discussion of the Relationship between GCNs and Graph Spectrum

Our spectral attack is based on the fact that the spectral filters are the fundamental building blocks for graph convolutions to process graph signals in the Fourier domain. Therefore, searching the graph perturbations in the direction that causes the most changes in the spectral filters, measured by eigenvalues of graph Laplacian, is expected to best disrupt graph convolutions. This is also supported by recent theoretical evidence in the field. Some recent literature has shown that the stability of GCN models is closely related to the eigenvalues of the graph Laplacian. For example, it is proved that the generalization gap of a single layer GCN model f_θ trained via T-step SGD is $\mathcal{O}(\lambda_n^{2T})$, where λ_n is the largest eigenvalue of graph Laplacian [178]. Meanwhile, Weinberger et al. [187] proved that a generalization estimate is *inversely* proportional to the second smallest eigenvalue of the graph Laplacian λ_2. These findings suggest that manipulating the graph by perturbing the eigenvalues can potentially aggravate the generalization gap of GCN, causing a larger generalization error.

2.6 Experiments

We performed extensive evaluations of the proposed spectral attack on four popularly used graph datasets, where we observed remarkable improvements in the attack's effectiveness. This section summarizes our experiment setup, performance on both evasion and poisoning attacks, and qualitative analysis on the perturbed graphs to study the effect of the spectral attack.

• **Datasets.** We evaluated the proposed attack on two citation network benchmark datasets, Cora [116] and Citeseer [152], as well as two social network datasets, Polblogs [3] and Blogcatalog [147]. Table 12

Table 12: Dataset statistics. D is the feature dimension. K is the number of classes.

Dataset	#Node	#Edge	Density	D	K
Cora	2,708	5,278	0.0014	1433	7
Citeseer	3,312	4,536	0.0008	3703	6
Polblogs	1,490	16,715	0.015	-	2
Blogcatalog	5,196	171,743	0.013	8189	6

Table 13: Average running time (in seconds) for 10 runs of evasion attack ($T = 100$, $\epsilon = 0.05$).

Dataset	Random	DICE	GF-Attack	**SPAC**	**SPAC-approx**
Cora	0.05	55.58	66.73	212.53	75.46
Citeseer	0.06	46.72	57.22	116.07	60.93
Polblogs	0.02	14.84	21.73	44.18	22.98
Blogcatalog	1.46	127.72	132.23	352.52	147.34

summarizes the statistics of these datasets. We followed the transductive semi-supervised node classification setup in [81], where only 20 sampled nodes per class were used for training, but the features and edges of all nodes were visible to the attacker during the training stage. The predictive accuracy of the trained classifier was evaluated on 1000 randomly selected test nodes. The evasion attacker can query the trained classifier, but cannot access the training nodes; the poisoning attacker can observe the training nodes, but cannot access the labels of the test nodes.

• **Baselines.** We compared the proposed attack model SPAC against three attacks in the black-box setting, and further verified the effectiveness of SPAC by combining it with five baselines in white-box setting[11]. Black-box baselines for both *evasion* and *poisoning* attack include: 1) **Random** directly attacks the graph structure by randomly flipping the edges; 2) **DICE** [186] is a heuristic method that deletes edges internally and connects nodes externally across class clusters; 3) **GF-Attack** [24] perturbs the structure by maximizing the loss of low-rank matrix approximation defined over small eigenvalues. We further evaluate the performance of SPAC combined with white-box attack baselines which include: 1) **PGD-CE** [193] is an *evasion* attack which maximizes the cross-entropy (CE) loss on the test nodes via projected gradient descent (PGD) algorithm [113]; 2) **PGD-C&W** [193] is an *evasion* attack which perturbs edges by minimizing the C&W score, which is the margin between the largest and the second-largest prediction score, defined by Carlini-Wagner attack [21]; 3) **Max-Min** [193] is a *poisoning* attack, which solves the bi-level optimization problem by iteratively generating structural perturbations (to maximize the cross-entropy loss) and retraining a surrogate victim model on the perturbed graph (to minimize the loss); 4) **Meta-Train** [209] is a *poisoning* attack which uses meta-gradients on the perturbation matrix to maximize the training classification loss; 5) **Meta-Self** [209] is a *poisoning* attack that extends Meta-Train to maximize the self-training loss on test nodes using the predicted labels;

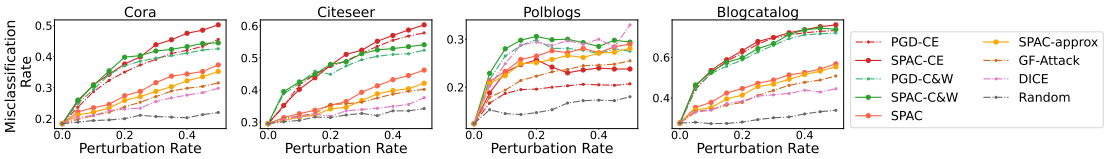
• **Variants of SPAC.** The proposed attack can be realized with exact and approximated spectral distance, which gives **SPAC** and **SPAC-approx**. We will compare their attack performance and running time. Meanwhile, adopting the objectives from white-box baselines to Eq (47) generates the following white-box attack variants: 1) **SPAC-CE** is an *evasion* attack that jointly maximizes the cross-entropy loss and spectral distance; 2) **SPAC-C&W** is an *evasion* attack combining the *negative* C&W score and spectral distance; 3) **SPAC-Min** extends Max-Min by maximizing the loss as in SPAC-CE for poisoning attack; 4) **SPAC-Train** includes the spectral distance to Meta-Train for the meta-gradient calculation; 5) **SPAC-Self** enhances the loss of Meta-Train by the spectral distance. The detailed objective for each variant is summarized in Appendix 3.3.

[11] We conducted the comparative experiments using DeepRobust Library [98].

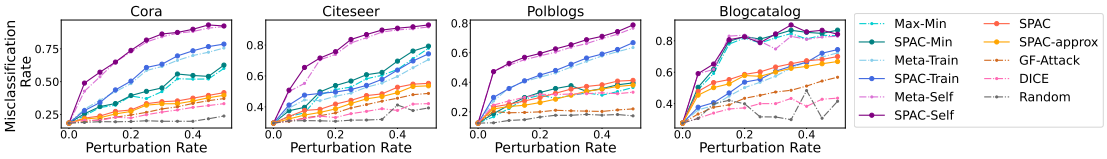
Table 14: Misclassification rate (%) with $\epsilon = 0.05$ for evasion (LEFT) and poisoning (RIGHT) attack.

Evasion	Cora	Citeseer	Polblogs	Blogcatalog
Clean	0.184	0.295	0.128	0.276
Random	0.189	0.301	0.153	0.280
DICE	0.205	0.308	0.202	0.329
GF-Attack	0.198	0.311	0.179	0.333
SPAC	**0.220**	**0.314**	**0.212**	**0.354**
SPAC-approx	0.212	0.305	0.208	0.341
PGD-CE	0.237	0.349	0.167	0.441
SPAC-CE	**0.255**	**0.352**	**0.188**	**0.458**
PGD-C&W	0.249	0.388	0.216	0.447
SPAC-C&W	**0.260**	**0.395**	**0.229**	**0.464**

Poison	Cora	Citeseer	Polblogs	Blogcatalog
Clean	0.184	0.295	0.128	0.276
Random	0.189	0.309	0.126	0.277
DICE	0.207	0.310	**0.246**	0.306
GF-Attack	0.195	0.306	0.202	0.334
SPAC	**0.222**	**0.338**	0.234	**0.478**
SPAC-approx	0.215	0.322	0.220	0.454
Max-Min	0.240	0.359	0.167	0.489
SPAC-Min	**0.255**	**0.375**	**0.188**	**0.504**
Meta-Train	**0.290**	0.392	0.274	0.360
SPAC-Train	0.285	**0.412**	**0.298**	**0.377**
Meta-Self	0.427	0.499	**0.478**	0.590
SPAC-Self	**0.489**	**0.508**	0.472	**0.599**



(a) Misclassification rate under different perturbation rates ϵ for evasion attack.



(b) Misclassification rate under different perturbation rates ϵ for poisoning attack.

Fig. 20: Misclassification rate under evasion (TOP) and poisoning (BOTTOM) attack with varying ϵ.

2.6.1 Structural Evasion Attack Performance In the evasion attack setting, we first trained a GCN classifier on the small training set V_0 with a clean graph $\widehat{\mathcal{G}} = \mathcal{G}$. Then the classifier was fixed, and the attackers generated edge perturbations based on the classifier's predictions on the test nodes. Table 14 summarizes the misclassification rates under $\epsilon = 0.05$, which allows 5% edges to be perturbed. An extensive comparison with different perturbation rates is provided in Figure 20(a), where the solid lines with darker color denote SPAC variants while the dashed lines with lighter color represent baseline attacks.

In the black-box setting, randomly flipping edges (Random) cannot effectively influence the classifier's overall performance. DICE provides an effective attack by leveraging the label information. GF-Attack undermines the performance of GCNs by attacking its low-rank approximation. Our methods, both SPAC and SPAC-approx, disrupt the overall spectral filters and achieve the largest misclassification rate. This shows the effectiveness of the proposed attack principle based on the spectral distance, which reveals the essence of vulnerability in graph convolutions.

Second in the white-box setting, SPAC-CE and SPAC-C&W stand in stark contrast to PGD-CE and PGD-C&W: we can observe a remarkable improvement introduced by SPAC in the misclassification rate. The evasion attack results confirm that maximizing the spectral distance can considerably disrupt the trained classifier by changing the graph frequencies in the Fourier domain and invalidating the spectral filters.

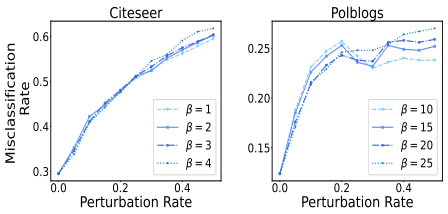


Fig. 21: Varying β for SPAC-CE attack.

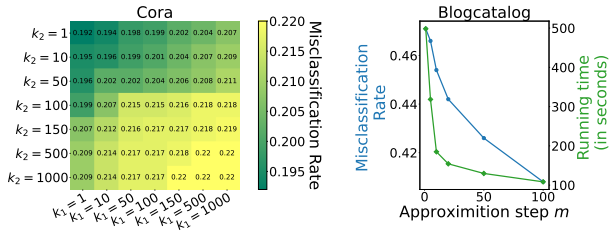


Fig. 22: Varying k_1, k_2 and m under SPAC-approx attack.

2.6.2 Structural Poisoning Attack Performance In the poisoning attack setting, we can only indirectly affect the classifier by perturbing the training graph structure. We generated the edge perturbations, and then used the poisoned structure to train the victim GCN model and reported its misclassification rate on test nodes in a clean graph. From Table 14 and Figure 20(b), we can again verify the effectiveness of the proposed spectral attack. Under the black-box setting, SPAC and SPAC-approx are the most effective attacks in most cases. Under the white-box setting, Max-Min only accesses training nodes to perturb the graph without querying test nodes. Meta-Train calculates the meta-gradient on training nodes to capture the change of loss after retraining the surrogate GCN model. Meta-Self instead does not use the training nodes, but only queries CGN's prediction scores on test nodes. Among baselines, the Meta-Self attack is shown to be the most effective, which is expected, because the current semi-supervised setting provides a much larger set of unlabeled nodes that can be fully used by Meta-Self. Overall, our attack based on the spectral distance still brought in a clear improvement to the misclassification rate across different datasets and attack methods.

We empirically evaluated the efficiency of SPAC and SPAC-approx in Table 13, which compares the average running time of 10 runs for evasion attack. Our proposed SPAC-approx can achieve a comparable efficiency as GF-Attack. Combining with the attack performance, SPAC-approx is verified to be an effective and efficient structural attack.

2.6.3 Analysis of SPAC Given the empirical effectiveness of the proposed attack strategy, we now analyze the sensitivity of hyper-parameters including k_1, k_2 and m for SPAC-approx and β for white-box setting. We also illustrate the behavior of SPAC in both Fourier and spatial domains.

• **Sensitivity of k_1, k_2 and m in SPAC-approx**. For SPAC-approx, the trade-off of its attack performance and efficiency is achieved by selecting k_1 low- and k_2 high-frequency components and by approximating the eigenvalue change for m steps. Figure 22 demonstrates such trade-off under SPAC-approx poisoning attack with budget $\epsilon = 0.05$. Left side shows the misclassification rate range when using different k_1 and k_2; right side compares the misclassification rate and running time when using different approximation step m. The result suggests that the attack performance does not dramatically drop with changed parameters, and we can achieve a good balance between attack effectiveness and efficiency.

• **Sensitivity of hyper-parameter β in white-box setting**. Figure 21 shows the performance of SPAC-CE under different settings of the coefficient parameter β. We can clearly observe that different β values lead to rather stable performance, which suggests the spectral distance term can be applied to real applications without the requirement of tedious hyper-parameter tuning.

• **Effect of SPAC in Fourier and spatial domain**. We are interested in investigating how the changes of the graph in the Fourier domain affect its spatial structure. To serve this purpose, we compared the output of the perturbed graphs from SPAC-CE and PGD-CE on Cora under budget $\epsilon = 0.4$ in Figure 23. The top

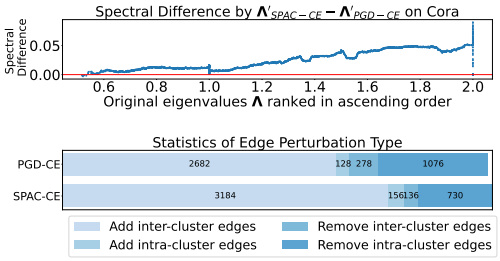
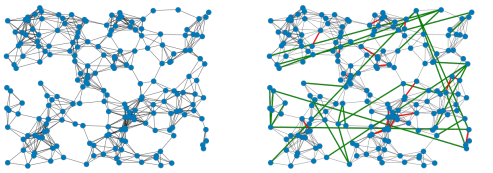


Fig. 23: Analysis on spectral changes (top) and spatial edge changes (bottom) between SPAC-CE and PGD-CE.



(a) Original graph (b) Perturbed graph by SPAC

Fig. 24: The edge perturbation generated by SPAC on a random geometric graph with $\epsilon = 0.05$. Green denotes edges added by the attack, while red marks removed edges.

plots the difference between eigenvalues of the normalized Laplacian matrix for the graph perturbed by SPAC-CE and the graph perturbed by PGD-CE. The x-axis shows the eigenvalues of the original graph. The bottom counts the number of different types of edge perturbations, where "inter-cluster" edges are those connecting nodes with different class labels and "intra-cluster" edges connect nodes with the same class label. We observe that SPAC-CE perturbed graph in a direction leading to larger high eigenvalues and smaller low eigenvalues, compared with PGD-CE. This spectral difference in the Fourier domain is also reflected in the spatial domain: 1) more edges are added than removed; 2) specifically, more inter-cluster edges were added while fewer inter-cluster edges were removed. To intuitively demonstrate the perturbations generated by SPAC, we applied SPAC to attack the random geometric graph with budget $\epsilon = 0.05$, and the perturbed graph is visualized in Figure 24. The green edges that are added by SPAC connect different node clusters, while red edges are removed within clusters. This shows that maximizing the spectral distance can modify the global connectivity of the graph: for example, SPAC-CE strengthened the connectivity between different clusters to confuse the classifier.

3 Conclusion

In this chapter, we focus on understanding and mitigating potential threats to the fairness and robustness of machine learning models, when the graph structure is incorporated in an undesired way. To cope with structural bias due to the influence from sensitive node attributes when the graph is generated, we propose a principled new way for learning unbiased node embeddings from biased and observed graph. The idea is to infer a bias-free graph where the influence from sensitive attributes is removed, and then learn the node embeddings from it. This new perspective motivates our design of UGE-W, UGE-R and their combined methods UGE-C. Extensive experiment results demonstrated strong debiasing effect from UGE as well as better unbiasedness-utility trade-offs in downstream applications. To study the vulnerability of graph embedding models to perturbed graph structure, we investigate an effective graph structural attack which disrupts graph spectral filters in the Fourier domain. We define the notion of spectral distance based on the eigenvalues of graph Laplacian to measure the disruption of spectral filters. We realize the attack by maximizing the spectral distance and propose an efficient approximation to reduce the time complexity brought by eigen-decomposition. The experiments demonstrate the remarkable effectiveness of the proposed attack in both black-box and white-box settings for both test-time evasion attacks and training-time poisoning attacks. Our qualitative analysis suggests the connection between the imposed spectral changes in the Fourier domain and the attack behavior in the spatial domain, which provides empirical evidence that maximizing spectral distance is an effective way to change the graph structural property and thus disturb the frequency components for graph filters to affect the learning of GCNs.

Chapter 4

Conclusion and Future Work

The ubiquity of graph-structured data calls for effective and trustworthy machine learning solutions that can make use of and learn to understand information represented in such a structured form. While the graph structure of data brings additional information and relational inductive bias for machine learning to better model entities, such structure also imposes complex dependency information among entities which could be misused to threaten the trustworthiness of machine learning models. This thesis demonstrate our understanding towards the computational question about the double-edged role of graph structure in machine learning. By elaborating the opportunities and the potential trustworthiness issues brought by graph structure, the outcome of this thesis can be applied to a variety of real-world problems.

In Chapter 2, we introduce our efforts in improving unsupervised machine learning with graph structural information, including explicit pairwise link, implicit cluster and global structural property summarized by graph spectrum. We treat the explicit edge as an indication of entity similarity, and jointly model network structure and textual content of user-generated data for the task of user representation learning. We design a generative model to integrate user representation learning with network and topic embedding. The learned user representations are interpretable and predictive, indicated by the performance improvement in many important tasks such as link prediction and expert recommendation. In the second work, we further discover an implicit hierarchy of groups and the nodes' affiliation to such groups. We align the group hierarchy with the convolutional layers in GCNs and design a joint node-level and group-level attention mechanism to aggregate the neighbor structure more accurately. Finally, we explore on improving contrastive learning with a graph augmentation scheme guided by maximizing the change of graph spectrum. These works demonstrates our insights of harnessing different types of information encoded in graph structure to improve unsupervised machine learning.

In Chapter 3, we study potential threats to the fairness and robustness of machine learning models, when the graph structure is incorporated in an undesired way. To mitigate structural bias, we propose a principled unbiased embedding model with theoretical justification to learn node embeddings from the observed graph by inferring a bias-free graph. This new perspective motivates our design of a class of unbiased models, including UGE-W, UGE-R and their combined methods UGE-C. We show a strong debiasing effect from UGE as well as better unbiasedness-utility trade-offs in downstream evaluations. Meanwhile, to study the vulnerability of graph structure, we investigate an effective graph structural attack by disrupting graph spectral filters in the Fourier domain. We realize the attack by maximizing the spectral distance between the original and perturbed graph, and propose an efficient approximation to reduce the time complexity brought by eigen-decomposition. Remarkable effectiveness of the proposed attack is achieved in both black-box and white-box settings for both test-time evasion attacks and training-time poisoning attacks. Our qualitative analysis also suggests the connection between the spectral changes in the Fourier domain and the attack behavior in the spatial domain.

1 Future Work

My thesis studies structural properties existing in interconnected systems, which is essential to understanding and exploiting the dependency structure among correlated instances (nodes). Based on our understanding of the informativeness, vulnerability, and biasedness of structure existing in relational data, we can further design scalable frameworks to advance trustworthy, responsible, and controllable machine learning in interconnected systems.

Large Scale Self-supervised Learning on Relational Data: Self-supervised learning empowers us to exploit a variety of labels crafted from data for free. Our previous work jointly modeling network structure and textural contents has shown the informativeness of unlabeled data [104, 50, 103]. In the past decade, we have observed the huge leap in AI brought by large-scale self-supervised learning models, including the record-high performances in ImageNet challenges and the success of pre-trained language models such as BERT and GPT-3. The OGB large-scale graph challenge is also recently published. Enabling effective and efficient self-supervised learning over large-scale relational data becomes demanding and influential on both industrial and scientific applications. However, training large-scale graph neural networks remains challenging: unlike common neural networks with training loss perfectly decomposed into individual terms on each sample, the objective of graph neural networks depends on a huge number of neighboring nodes especially when the model goes deep. To support distributed and federated self-supervised learning on large-scale relational data, it is of great significance to develop efficient graph processing algorithms (e.g., segmentation, sampling and augmentation) and customized optimizers that allow efficient (adversarial and self-supervised) training for widely used graph neural networks.

Adversarial Robustness on Discrete Data: Discrete data, such as text, graph, categorical features, pervasively exist in real life. Machine learning models dealing with such data can also be attacked by perturbations defined in discrete space. For example in graph based tasks, an attacker may fool the graph model by injecting unnoticeable changes to the graph structure (deleting or adding edges); in fake news detection tasks, an attacker may mislead a detector by simply changing the word “pleased” into “delighted”. Studying the adversarial robustness of models training on discrete data is important but challenging: finding the optimal discrete perturbation is intrinsically an NP-hard combinatorial optimization task; gradient-based methods cannot be directly applied in discrete space. It still remains an open problem how to perform effective and efficient searches towards the optimal solutions. We has already shown the effectiveness of random sampling to discretize the continuous gradients on graphs [102]. To extend my research for discrete data, the key is to properly define the perturbation distance and decision boundary in the discrete space. It becomes essential to develop efficient search strategies to systemically explore adversarial robustness and utilize the knowledge gained to further build robust models on discrete data.

Learnability Control of Models and Data: More and more personal data are shared with many organizations, including online social platforms to communicate with friends, healthcare providers to benefit from personalized care and retailers to receive personal recommendations. Misusing of personal data raises both fairness and privacy issue, and it is a challenging problem to understand and control the influence of data instances once the data or model is released. The difficulty lies in both model and data sides: for the model side, it is unclear how it can remove the influence from a subset of data when users request deleting their sensitive data from a serving model. This problem is exacerbated when deep learning models are known to memorize some of the training data. Machine unlearning **[17]** is proposed to develop an algorithm that takes as input a trained model and outputs a new one with requested data removed. For the data side, it is not obvious how to prevent unauthorized exploitation of personal data when realizing the dataset. Unlearnable examples **[66]** is proposed to attack the learnability such that the dataset becomes unlearnable by machine learning models with negligible data modification. Given these prior attempts on learnability control of model and data, the next step is to study the learnability problem by investigating

the dependency correlation between data instances and their influence on model prediction, such that the protected data can be isolated and further removed.

Human-Out-of-the-Loop Machine learning: Machine learning certainly liberates our hands and brains in both scientific and industrial endeavors. However, the current machine learning algorithms are not human-out-of-the-loop: we still rely on domain knowledge and human experiences in designing and optimizing machine learning systems. And in many cases, we still need humans labeling the dataset for machine learning algorithms to work. Any significant advances in automating such processes can result in significant productivity improvements and innovations for a wide range of domains. Therefore, one future research direction is to achieve human-out-of-the-loop machine learning. Such a goal is ambitious but not impracticable, and progress has been made in areas such as automated machine learning (AutoML). However, such techniques are still far from mature and a lot of challenges are yet to be addressed. For example, how to make sure the automation process itself is robust, trustworthy and efficient to various real-world problems. Also in cases where human labeling is not available, how to automatically design self-supervised training schemes and pretexts.

Human-level Trustworthy and Responsible Machine Learning: One long-term future trend is to develop human-level trustworthy and accountable machine learning methods without the cost of accuracy drop. Current robust training models [202] and my work in fair model [184] usually suffer from such trade-off while human eyes can be accurate, robust and fair. Such a difference suggests that we still have not figured out the right way to train trustworthy and responsible models. Fortunately, recent studies show that adversarial training could actually help improve natural accuracy in natural language processing tasks, which implies that such a trade-off is not inevitable. To further understand such trade-off effects and to finally achieve human-level robustness and fairness, I aim to study the relationship between model prediction and other factors such as the inductive bias of the model as well as dataset properties.

2 Broader Impact

The proposed research tries to solve several key challenges of exploiting graph structure in machine learning models. On one hand, the structure brings additional information that can advance machine learning; on the other hand, the structure could be biased and perturbed in an undesired way which may threaten the accountability and trustworthiness of machine learning. The significance of proposed research is that we provide a principled way to take the advantage of structure while mitigating the potential pitfalls for a more effective and trustworthy machine learning. The proposed approaches are verified by extensive empirical evaluation, and some of them are supported by rigorous theoretical analysis. This line of research will establish a new foundation of machine learning with graph structure, and further trigger new development of learning paradigm to unleash the power of relationship and graph structure in an ethical, interpretable and trustworthy way. More generally, the proposed research can benefit a wide spectrum of important real-world applications, including personalized recommendation, user modeling, text understanding, cybersecurity and many more.

Bibliography

[1] Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. (2019). Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR.

[2] Acharya, A., Teffer, D., Henderson, J., Tyler, M., Zhou, M., and Ghosh, J. (2015). Gamma process poisson factorization for joint modeling of network and documents. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 283–299. Springer.

[3] Adamic, L. A. and Glance, N. (2005). The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43.

[4] Agarwal, C., Lakkaraju, H., and Zitnik, M. (2021). Towards a unified framework for fair and stable graph representation learning. In *Proceedings of Conference on Uncertainty in Artificial Intelligence, UAI*.

[5] Airoldi, E. M., Blei, D., Fienberg, S., and Xing, E. (2008). Mixed membership stochastic blockmodels. *Advances in neural information processing systems*, 21.

[6] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[7] Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591.

[8] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[9] Berk, R., Heidari, H., Jabbari, S., Kearns, M., and Roth, A. (2021). Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 50(1):3–44.

[10] Bianchi, F. M., Grattarola, D., and Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pages 874–883. PMLR.

[11] Bielak, P., Kajdanowicz, T., and Chawla, N. V. (2021). Graph barlow twins: A self-supervised representation learning framework for graphs. *arXiv preprint arXiv:2106.02466*.

[12] Blei, D. and Lafferty, J. (2006). Correlated topic models. *Advances in neural information processing systems*, 18:147.

[13] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[14] Bojchevski, A. and Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pages 695–704. PMLR.

[15] Bose, A. J. and Hamilton, W. L. (2019). Compositional fairness constraints for graph embeddings.

[16] Bourigault, S., Lagnier, C., Lamprier, S., Denoyer, L., and Gallinari, P. (2014). Learning social network embeddings for predicting information diffusion. In *Proceedings of the 7th ACM WSDM*, pages 393–402. ACM.

[17] Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2021). Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE.

[18] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[19] Buyl, M. and De Bie, T. (2020). Debayes: a bayesian method for debiasing network embeddings. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1220–1229.

[20] Calmon, F. P., Wei, D., Vinzamuri, B., Ramamurthy, K. N., and Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3995–4004.

[21] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

[22] Chang, H., Rong, Y., Xu, T., Bian, Y., Zhou, S., Wang, X., Huang, J., and Zhu, W. (2021a). Not all low-pass filters are robust in graph convolutional networks. *Advances in Neural Information Processing Systems*, 34.

[23] Chang, H., Rong, Y., Xu, T., Huang, W., Sojoudi, S., Huang, J., and Zhu, W. (2021b). Spectral graph attention network with fast eigen-approximation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2905–2909.

[24] Chang, H., Rong, Y., Xu, T., Huang, W., Zhang, H., Cui, P., Zhu, W., and Huang, J. (2020). A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3389–3396.

[25] Chang, J. and Blei, D. (2009). Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88.

[26] Chen, H., Perozzi, B., Hu, Y., and Skiena, S. (2018a). Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[27] Chen, H., Sun, M., Tu, C., Lin, Y., and Liu, Z. (2016). Neural sentiment classification with user and product attention. In *Proceedings of the 2016 EMNLP*, pages 1650–1659.

[28] Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., and Xuan, Q. (2018b). Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*.

[29] Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266.

[30] Chiplunkar, A., Kapralov, M., Khanna, S., Mousavifar, A., and Peres, Y. (2018). Testing graph clusterability: Algorithms and lower bounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 497–508. IEEE.

[31] Chouldechova, A. (2016). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments.

[32] Chung, F. and Lu, L. (2002). The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882.

[33] Chung, F. R. and Graham, F. C. (1997). *Spectral graph theory*. American Mathematical Soc.

[34] Clauset, A., Moore, C., and Newman, M. E. (2006). Structural inference of hierarchies in networks. In *ICML Workshop on Statistical Network Analysis*, pages 1–13. Springer.

[35] Clauset, A., Moore, C., and Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101.

[36] Dai, E. and Wang, S. (2020). Learning fair graph neural networks with limited and private sensitive attribute information. *arXiv preprint arXiv:2009.01454*.

[37] Dai, E. and Wang, S. (2021). Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 680–688.

[38] Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. (2018). Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR.

[39] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852.

[40] Donnat, C., Zitnik, M., Hallac, D., and Leskovec, J. (2018). Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1320–1329.

[41] Du, L., Lu, Z., Wang, Y., Song, G., Wang, Y., and Chen, W. (2018). Galaxy network embedding: A hierarchical community structure preserving approach. In *IJCAI*, pages 2079–2085.

[42] Duong, C. T., Hoang, D., Le Ba, T. G., Le Cong, T., Yin, H., Weidlich, M., Nguyen, Q. V. H., and Aberer, K. (2019). Unsupervised learning of node embeddings by detecting communities.

[43] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

[44] Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. (2020). All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177.

[45] Feng, S., Jing, B., Zhu, Y., and Tong, H. (2022). Adversarial graph contrastive learning with information regularization. *arXiv preprint arXiv:2202.06491*.

[46] Fischer, G. (2001). User modeling in human–computer interaction. *User modeling and user-adapted interaction*, 11(1-2):65–86.

[47] Godsil, C. D. (1981). On the full automorphism group of a graph. *Combinatorica*, 1(3):243–256.

[48] Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276.

[49] Gong, L., Al Boni, M., and Wang, H. (2016). Modeling social norms evolution for personalized sentiment classiofication. In *Proceedings of the 54th ACL*, volume 1, pages 855–865.

[50] Gong, L., Lin, L., Song, W., and Wang, H. (2020). Jnet: Learning user representations via joint network embedding and topic embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 205–213.

[51] Gong, L. and Wang, H. (2018). When sentiment analysis meets social network: A holistic user behavior modeling in opinionated data. In *Proceedings of the 24th ACM SIGKDD*, pages 1455–1464. ACM.

[52] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

[53] Guo, J., Li, S., Zhao, Y., and Zhang, Y. (2022). Learning robust representation through graph adversarial contrastive learning. *arXiv preprint arXiv:2201.13025*.

[54] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034.

[55] Hammond, D. K., Gur, Y., and Johnson, C. R. (2013). Graph diffusion distance: A difference measure for weighted graphs based on the graph laplacian exponential kernel. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 419–422.

[56] Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

[57] Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29:3315–3323.

[58] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

[59] Hassani, K. and Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR.

[60] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

[61] Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

[62] Hu, F., Zhu, Y., Wu, S., Wang, L., and Tan, T. (2019). Hierarchical graph convolutional networks for semi-supervised node classification. *arXiv preprint arXiv:1902.06667*.

[63] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020a). Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

[64] Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2020b). Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

[65] Hu, Z., Dong, Y., Wang, K., Chang, K.-W., and Sun, Y. (2020c). Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[66] Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. (2021). Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*.

[67] Jiao, Y., Xiong, Y., Zhang, J., Zhang, Y., Zhang, T., and Zhu, Y. (2020). Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 222–231. IEEE.

[68] Jiao, Y., Xiong, Y., Zhang, J., and Zhu, Y. (2019). Collective link prediction oriented network embedding with hierarchical graph attention. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 419–428.

[69] Jin, D., Li, B., Jiao, P., He, D., and Zhang, W. (2019). Network-specific variational auto-encoder for embedding in attribute networks. In *IJCAI*, pages 2663–2669.

[70] Jin, M., Chang, H., Zhu, W., and Sojoudi, S. (2021a). Power up! robust graph convolutional network against evasion attacks based on graph powering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[71] Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., and Tang, J. (2020). Self-supervised learning on graphs: Deep insights and new direction.

[72] Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., and Tang, J. (2021b). Adversarial attacks and defenses on graphs. *SIGKDD Explor. Newsl.*, 22(2):19–34.

[73] Johnson, B. R. and Linksvayer, T. A. (2010). Deconstructing the superorganism: social physiology, groundplans, and sociogenomics. *The Quarterly Review of Biology*, 85(1):57–79.

[74] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

[75] Kahale, N. (1995). Eigenvalues and expansion of regular graphs. *J. ACM*, 42(5):1091–1106.

[76] Kamishima, T., Akaho, S., Asoh, H., and Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. In Flach, P. A., De Bie, T., and Cristianini, N., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 35–50, Berlin, Heidelberg. Springer Berlin Heidelberg.

[77] Kang, B., Lijffijt, J., and Bie, T. D. (2018). Conditional network embeddings.

[78] Kenlay, H., Thanou, D., and Dong, X. (2020). On the stability of polynomial spectral graph filters. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5350–5354. IEEE.

[79] Kenlay, H., Thanou, D., and Dong, X. (2021). Interpretable stability bounds for spectral graph filters. In *International conference on machine learning*, pages 5388–5397. PMLR.

[80] Kesebir, S. (2012). The superorganism account of human sociality: How and when human groups are like beehives. *Personality and Social Psychology Review*, 16(3):233–261.

[81] Kipf, T. N. and Welling, M. (2017a). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[82] Kipf, T. N. and Welling, M. (2017b). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[83] Kloek, T. and van Dijk, H. K. (1978). Bayesian estimates of equation system parameters: An application of integration by monte carlo. *Econometrica*, 46(1):1–19.

[84] Kobsa, A. (2001). Generic user modeling systems. *User modeling and user-adapted interaction*, 11(1-2):49–63.

[85] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8):30–37.

[86] Kulik, C. T., Robert Jr, L., et al. (2000). Demographics in service encounters: effects of racial and gender congruence on perceived fairness. *Social Justice Research*, 13(4):375–402.

[87] La Fond, T. and Neville, J. (2010). Randomization tests for distinguishing social influence and homophily effects. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 601–610, New York, NY, USA. Association for Computing Machinery.

[88] Lecky, P. (1945). Self-consistency; a theory of personality.

[89] Lee, J. R., Gharan, S. O., and Trevisan, L. (2014). Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):1–30.

[90] Lei, L., Li, X., and Lou, X. (2020). Consistency of spectral clustering on hierarchical stochastic block models. *arXiv preprint arXiv:2004.14531*.

[91] Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. (2010). Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11:985–1042.

[92] Levie, R., Isufi, E., and Kutyniok, G. (2019). On the transferability of spectral graph filters. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pages 1–5. IEEE.

[93] Li, C.-T. and Lin, H.-Y. (2020). Structural hierarchy-enhanced network representation learning. *Applied Sciences*, 10(20):7214.

[94] Li, J., Rong, Y., Cheng, H., Meng, H., Huang, W., and Huang, J. (2019). Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*, pages 972–982.

[95] Li, J., Xie, T., Liang, C., Xie, F., He, X., and Zheng, Z. (2021a). Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering*.

[96] Li, M. M., Huang, K., and Zitnik, M. (2021b). Representation learning for networks in biology and medicine: Advancements, challenges, and opportunities. *CoRR*, abs/2104.04883.

[97] Li, Q., Han, Z., and Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[98] Li, Y., Jin, W., Xu, H., and Tang, J. (2020). Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*.

[99] Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031.

[100] Lin, L., Blaser, E., and Wang, H. (2021a). Graph embedding with hierarchical attentive membership. *arXiv preprint arXiv:2111.00604*.

[101] Lin, L., Blaser, E., and Wang, H. (2021b). Graph structural attack by spectral distance.

[102] Lin, L., Blaser, E., and Wang, H. (2022). Graph structural attack by perturbing spectral distance. *Proceedings of the 28th ACM SIGKDD international conference on knowledge discovery & data mining, KDD*.

[103] Lin, L., Gong, L., and Wang, H. (2019). Learning personalized topical compositions with item response theory. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 609–617. ACM.

[104] Lin, L. and Wang, H. (2020). Graph attention networks over edge content-based channels. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1819–1827.

[105] Liu, P., Zhang, L., and Gulla, J. A. (2019). Real-time social recommendation based on graph embedding and temporal context. *International Journal of Human-Computer Studies*, 121:58–72.

[106] Long, Q., Jin, Y., Song, G., Li, Y., and Lin, W. (2020). Graph structural-topic neural network. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1065–1073.

[107] Long, Q., Wang, Y., Du, L., Song, G., Jin, Y., and Lin, W. (2019). Hierarchical community structure preserving network embedding: A subspace approach. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 409–418.

[108] Ma, J., Cui, P., Wang, X., and Zhu, W. (2018a). Hierarchical taxonomy aware network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1920–1929.

[109] Ma, J., Ding, S., and Mei, Q. (2020). Towards more practical adversarial attacks on graph neural networks. *Advances in Neural Information Processing Systems*, 33.

[110] Ma, Y., Ren, Z., Jiang, Z., Tang, J., and Yin, D. (2018b). Multi-dimensional network embedding with hierarchical structure. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 387–395.

[111] Ma, Y., Wang, S., Derr, T., Wu, L., and Tang, J. (2021). Graph adversarial attack via rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1161–1169. Association for Computing Machinery.

[112] Madras, D., Creager, E., Pitassi, T., and Zemel, R. (2018). Learning adversarially fair and transferable representations.

[113] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

[114] Magnus, J. R. (1985). On differentiating eigenvalues and eigenvectors. *Econometric theory*, 1(2):179–191.

[115] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

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

[117] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Review of Sociology*, 27:415–444.

[118] Mei, Q., Cai, D., Zhang, D., and Zhai, C. (2008). Topic modeling with network regularization. In *Proceedings of the 17th WWW*, pages 101–110. ACM.

[119] Mernyei, P. and Cangea, C. (2020). Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*.

[120] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[121] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

[122] Miller, B. A., Beard, M. S., and Bliss, N. T. (2011). Matched filtering for subgraph detection in dynamic networks. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pages 509–512. IEEE.

[123] Mnih, A. and Salakhutdinov, R. R. (2008). Probabilistic matrix factorization. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

[124] Mohar, B., Alavi, Y., Chartrand, G., and Oellermann, O. (1991). The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12.

[125] Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

[126] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

[127] Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30:6338–6347.

[128] Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1105–1114, New York, NY, USA. Association for Computing Machinery.

[129] Palowitch, J. and Perozzi, B. (2020). Monet: Debiasing graph embeddings via the metadata-orthogonal training unit.

[130] Park, C., Yang, C., Zhu, Q., Kim, D., Yu, H., and Han, J. (2020). Unsupervised differentiable multi-aspect network embedding. *arXiv preprint arXiv:2006.04239*.

[131] Parlett, B. N. and Scott, D. S. (1979). The lanczos algorithm with selective orthogonalization. *Mathematics of computation*, 33(145):217–238.

[132] Peixoto, T. P. (2014). Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047.

[133] Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., and Huang, J. (2020). Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270.

[134] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

[135] Pfeiffer, J. J., Moreno, S., La Fond, T., Neville, J., and Gallagher, B. (2014). Attributed graph models: Modeling network structure with correlated attributes. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 831–842, New York, NY, USA. Association for Computing Machinery.

[136] Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., and Tucker, G. (2019). On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR.

[137] Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160.

[138] Rahman, T., Surma, B., Backes, M., and Zhang, Y. (2019). Fairwalk: Towards fair graph embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3289–3295. International Joint Conferences on Artificial Intelligence Organization.

[139] Raman, M., Chan, A., Agarwal, S., Wang, P., Wang, H., Kim, S., Rossi, R., Zhao, H., Lipka, N., and Ren, X. (2021). Learning to deceive knowledge graph augmented models via targeted perturbation. In *International Conference on Learning Representations*.

[140] Rampini, A., Pestarini, F., Cosmo, L., Melzi, S., and Rodola, E. (2021). Universal spectral adversarial attacks for deformable shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3216–3226.

[141] Rellich, F. and Berkowitz, J. (1969). *Perturbation theory of eigenvalue problems*. CRC Press.

[142] Rendle, S. (2010). Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE.

[143] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback.

[144] Rogers, L. C. (1970). Derivatives of eigenvalues and eigenvectors. *AIAA journal*, 8(5):943–944.

[145] Rong, Y., Huang, W., Xu, T., and Huang, J. (2020). Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

[146] Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th conference on UAI*, pages 487–494. AUAI Press.

[147] Rossi, R. and Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[148] Rozemberczki, B., Allen, C., and Sarkar, R. (2019). Multi-scale attributed node embedding.

[149] Sandryhaila, A. and Moura, J. M. (2013). Discrete signal processing on graphs: Graph fourier transform. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6167–6170. IEEE.

[150] Schaub, M. T. and Segarra, S. (2018). Flow smoothing and denoising: Graph signal processing in the edge-space. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 735–739. IEEE.

[151] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

[152] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.

[153] Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

[154] Shen, X., Tan, B., and Zhai, C. (2005). Implicit user modeling for personalized search. In *Proceedings of the 14th ACM CIKM*, pages 824–831. ACM.

[155] Song, Y., Shu, R., Kushman, N., and Ermon, S. (2018). Constructing unrestricted adversarial examples with generative models. *Advances in Neural Information Processing Systems*, 31:8312–8323.

[156] Srinivasan, B. and Ribeiro, B. (2019). On the equivalence between positional node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452*.

[157] Stewart, G. W. (1990). Matrix perturbation theory.

[158] Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. (2019). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*.

[159] Sun, L., Dou, Y., Yang, C., Wang, J., Yu, P. S., He, L., and Li, B. (2018). Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*.

[160] Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., and Honavar, V. (2020). *Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach*, page 673–683. Association for Computing Machinery, New York, NY, USA.

[161] Suresh, S., Li, P., Hao, C., and Neville, J. (2021). Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34.

[162] Takac, L. and Zabovsky, M. (2012). Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*.

[163] Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., and Li, P. (2011). User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD*, pages 1397–1405. ACM.

[164] Tang, D., Qin, B., and Liu, T. (2015a). Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd ACL*, volume 1, pages 1014–1023.

[165] Tang, J., Chang, Y., and Liu, H. (2014). Mining social media with social theories: A survey. *ACM SIGKDD Explorations Newsletter*, 15(2):20–29.

[166] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015b). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077.

[167] Tang, L. and Liu, H. (2009a). Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD*, pages 817–826. ACM.

[168] Tang, L. and Liu, H. (2009b). Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1107–1116.

[169] Thakoor, S., Tallec, C., Azar, M. G., Munos, R., Veličković, P., and Valko, M. (2021). Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

[170] Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020). What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839.

[171] Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

[172] Tuckey, M. R. and Brewer, N. (2003). The influence of schemas, stimulus ambiguity, and interview schedule on eyewitness memory over time. *Journal of Experimental Psychology: Applied*, 9(2):101.

[173] Van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807.

[174] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018a). Graph Attention Networks. *International Conference on Learning Representations*.

[175] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018b). Graph Attention Networks. *International Conference on Learning Representations*. accepted as poster.

[176] Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep graph infomax. *ICLR (Poster)*, 2(3):4.

[177] Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep Graph Infomax. In *International Conference on Learning Representations*.

[178] Verma, S. and Zhang, Z.-L. (2019). Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1539–1548.

[179] Wai, H.-T., Segarra, S., Ozdaglar, A. E., Scaglione, A., and Jadbabaie, A. (2018). Community detection from low-rank excitations of a graph filter. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4044–4048. IEEE.

[180] Wang, B. and Gong, N. Z. (2019). Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2023–2040.

[181] Wang, B., Zhou, T., Lin, M., Zhou, P., Li, A., Pang, M., Fu, C., Li, H., and Chen, Y. (2020). Evasion attacks to graph neural networks via influence function. *arXiv preprint arXiv:2009.00203*.

[182] Wang, C. and Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD*, pages 448–456. ACM.

[183] Wang, H., Lu, Y., and Zhai, C. (2011). Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD*, pages 618–626. ACM.

[184] Wang, N., Lin, L., Li, J., and Wang, H. (2021). Unbiased graph embedding with biased graph observations. *arXiv preprint arXiv:2110.13957*.

[185] Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. (2017). Community preserving network embedding. In *AAAI*, pages 203–209.

[186] Waniek, M., Michalak, T. P., Wooldridge, M. J., and Rahwan, T. (2018). Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147.

[187] Weinberger, K. Q., Sha, F., Zhu, Q., and Saul, L. K. (2007). Graph laplacian regularization for large-scale semidefinite programming. In *Advances in neural information processing systems*, pages 1489–1496.

[188] White, R. W., Chu, W., Hassan, A., He, X., Song, Y., and Wang, H. (2013). Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd WWW*, pages 1411–1420. ACM.

[189] Woodworth, B., Gunasekar, S., Ohannessian, M. I., and Srebro, N. (2017). Learning non-discriminatory predictors. In Kale, S. and Shamir, O., editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1920–1953. PMLR.

[190] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019a). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

[191] Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. (2019b). Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 4816–4823. AAAI Press.

[192] Xie, M., Yin, H., Wang, H., Xu, F., Chen, W., and Wang, S. (2016). Learning graph-based poi embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 15–24.

[193] Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. (2019a). Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[194] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019b). How powerful are graph neural networks? In *International Conference on Learning Representations*.

[195] Yan, Z., Ma, T., Gao, L., Tang, Z., and Chen, C. (2021). Link prediction with persistent homology: An interactive view. In *International Conference on Machine Learning*, pages 11659–11669. PMLR.

[196] Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. (2015). Network representation learning with rich text information. In *IJCAI*, pages 2111–2117.

[197] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pages 4800–4810.

[198] You, Y., Chen, T., Shen, Y., and Wang, Z. (2021). Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR.

[199] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020a). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823.

[200] You, Y., Chen, T., Wang, Z., and Shen, Y. (2020b). When does self-supervision help graph convolutional networks? In *international conference on machine learning*, pages 10871–10880. PMLR.

[201] Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. (2013). Learning fair representations. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333, Atlanta, Georgia, USA. PMLR.

[202] Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. (2019). Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*.

[203] Zhang, H., Zhou, J., and Li, R. (2020). Enhanced unsupervised graph embedding via hierarchical graph convolution network. *Mathematical Problems in Engineering*, 2020.

[204] Zhang, M. and Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems*, 31:5165–5175.

[205] Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. (2020). Deep Graph Contrastive Representation Learning. In *ICML Workshop on Graph Representation Learning and Beyond*.

[206] Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. (2021). Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, WWW '21, page 2069–2080, New York, NY, USA. Association for Computing Machinery.

[207] Zitnik, M. and Leskovec, J. (2017). Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198.

[208] Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856.

[209] Zügner, D. and Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*.

Chapter 5

Appendix

1 Additional Background and Result of Chapter 2, Section 3

1.1 A Review of Graph Spectrum

Graph spectrum plays a significant role in analyzing graph property (e.g., connectivity, cluster structure, diameter etc.) and is the foundation of spectral filters in GNNs. This motivates us to guide our proposed topology augmentation method using graph spectrum.

• **Graph Spectrum and Graph Property.** The graph spectrum summarizes important properties related to a graph's global structure, which has been studied in graph spectral theory. We list some widely discussed properties revealed by graph spectrum to support our design: graph spectrum can be used as a comprehensive proxy for capturing graph properties in GCL.

- *Algebraic connectivity* [33], also known as Fiedler eigenvalue, of a graph is the second-smallest eigenvalue (counting multiple eigenvalues separately) of the Laplacian matrix. This eigenvalue is greater than 0 if and only if the graph is a connected graph. A corollary is that the number of times 0 appears as an eigenvalue in the Laplacian is the number of connected components in the graph. The magnitude of this value reflects how well connected the overall graph is.
- *Diameter* of a graph can be upper and lower bounded from its spectrum [33]. If the graph has r distinct eigenvalues, its diameter d is at most $r-1$. Meanwhile, if the graph has m edges and n nodes, we can bound the diameter by the first and second smallest non-zero eigenvalues as $1/2m\lambda_1 \geq d \geq 4/n\lambda_2$. For all $k \geq 2$, we also have $d \leq k \log n/\lambda_k$.
- *Clusterability* of a graph reveals how easy it is to partition the graph, which can be captured by the differences between the smallest successive eigenvalues of connected graphs. The difference between the first two eigenvalues, often referred to as the spectral gap, upper and lower bounds the graph expansion and conductance by the Cheeger inequality [75]. Nevertheless, analogous results also hold for higher-order eigenvalues [89].
- *Diffusion distance* [55] between two nodes v_i and v_j can be defined as $\mathcal{D}(v_i, v_j) = \|[\phi(\mathbf{L})]_{i,:} - [\phi(\mathbf{L})]_{j,:}\|_2^2 = \sum_{l=1}^{n} \phi(\lambda_l)^2(\mathbf{u}_l[i] - \mathbf{u}_l[j])^2$, where $\phi(\mathbf{L}) = \mathbf{U}\phi(\mathbf{\Lambda})\mathbf{U}^\top$ and $\phi(x)$ is a decreasing kernel function such as $\phi(x) = e^{-tx}$. Therefore, a map that separates nodes with a specific diffusion distance is obtained when embedding graph nodes using eigenvectors.

• **Graph Spectrum and Spectral Filter.** By viewing graph embedding models from a signal processing perspective, the normalized Laplacian $\mathbf{L}_{\text{norm}}$ serves as a graph shift operator and defines the frequency

domain of a graph. As a result, its eigenvectors $\mathbf{U}$ can be considered as the graph Fourier bases, and the eigenvalues $\mathbf{\Lambda}$ (a.k.a., graph spectrum) correspond to the frequency components. Take one column of node features $\mathbf{X}$ as an example of graph signal, which can be compactly represented as $\mathbf{x} \in \mathbb{R}^n$. The graph Fourier transform of graph signal $\mathbf{x}$ is given by $\widehat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$ and the inverse graph Fourier transform is then $\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}}$. The graph signals in the Fourier domain are filtered by amplifying or attenuating the frequency components $\mathbf{\Lambda}$.

At the essence of GNNs is the *spectral convolution*, which can be defined as the multiplication of a signal vector $\mathbf{x}$ with a spectral filter g_ϕ in the Fourier domain [39]:

$$g_\phi(\mathbf{L}) \star \mathbf{x} = g(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top)\mathbf{x} = \mathbf{U}g_\phi(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{x} \tag{48}$$

The filter g_ϕ defines a smooth transformation function of the graph spectrum. One can apply a spectral filter and graph Fourier transformation to manipulate graph signals in various way, such as smoothing and denoising [150], abnormally detection [122] and clustering [179]. Here we show how the spectral convlution is defined in two popular GNNS used in our experiments: GCN [82] and GIN [194].

The vanilla GCN [82] is a first-order approximation of Chebyshev polynomial filter [56] with $g_\phi(\mathbf{\Lambda}) = (2 - \mathbf{\Lambda})\phi$, and the corresponding convolution for d-dimensional signal $\mathbf{X}$ is:

$$g_\phi^{\text{GCN}}(\mathbf{L}) \star \mathbf{X} = \mathbf{U}(2 - \mathbf{\Lambda})\mathbf{U}^\top\mathbf{X}\mathbf{\Phi} = (\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{X}\mathbf{\Phi} = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Phi} \tag{49}$$

where $\mathbf{\Phi} \in \mathbf{R}^{d \times d'}$ is the matrix of spectral filter parameters, and by adding self-loop $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$, a renormalization trick $\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \rightarrow \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}$ is applied. GIN [194] with equal discriminative power as WL test designs spectral convolution as:

$$g_\phi^{\text{GIN}}(\mathbf{L}) \star \mathbf{X} = \mathbf{U}(2 + \epsilon - \mathbf{\Lambda})\mathbf{U}^\top\mathbf{X}\mathbf{\Phi} = (\mathbf{I}_n(1 + \epsilon) + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{X}\mathbf{\Phi} \tag{50}$$

where ϵ can be a learnable parameter or a fixed scalar. Since the spectral filters $g_\phi(\mathbf{\Lambda})$ are the key in graph convolutions to encode graph signals that are transformed in the Fourier domain. The output of the spectral filters is then transformed back to the spatial domain to generate node representations. Therefore, we aim to augment graphs to influence the graph spectrum and the filtered graph signals, such that the encoder with altered spectral filters is encouraged to stay invariant to such perturbations through GCL.

1.2 Pre-analysis Experiment Setup of Figure 13

We now introduce the detailed information to reproduce Figure 13 on Cora. This experiment is to show that uniformly random edge perturbation adopted in many GCL methods is not effective enough to reveal essential graph properties, described by graph spectrum. Since graph spectrum is closely related to graph properties such as clusterability (as discussed in Appendix 1.1), in contrast to the uniform edge perturbation, we created a node cluster based strategy as follows: We first grouped the edges among nodes by whether the end nodes belong to the same cluster, treating nodes' class labels as their clusters. For intra-cluster edges, we assign a larger removing probability, while for inter-cluster edges we assign a smaller removing probability. Note that in expectation, we remove the same amount of edges as the uniformly *random* strategy, but allocate different probabilities to these two groups of edges. We should note the purpose of this experiment is only to demonstrate the impact of graph spectrum for GCL, and we used the class label of nodes as a proxy about the graph spectrum (due to the relation between clusterability and graph spectrum). Our proposed solution GCL-TAGS is fully unsupervised, i.e., it does *not* depend on the node labels at all.

Table 15: Node classification dataset. The metric is *accuracy*.

Data Name	#Nodes	#Edges	#Features	#Classes	Cluster Coefficient
Cora	2,708	5,278	1,433	7	0.2407
Citeseer	3,327	4,552	3,703	6	0.1415
PubMed	19,717	44,324	500	3	0.0602
Wiki-CS	11,701	215,863	300	10	0.4527
Amazon-Computers	13,752	245,861	767	10	0.3441
Amazon-Photos	7,650	119,081	745	8	0.4040
Coauthor-CS	18,333	81,894	6,805	15	0.3425

Specifically, an edge removing ratio σ indicated by the x-axis of Figure 13 represents the augmentation strength: for an input graph with m edges, we remove $\sigma \cdot m$ edges to generate an augmented graph. For the uniformly random augmentation method (*Uniform*), each edge is assigned an equal removing probability as σ; for the cluster-based augmentation heuristic (*Clustered*), given m_{inter} inter-cluster edges and $m_{\text{intra}} = m - m_{\text{inter}}$ intra-cluster edges, we increase the removing probability of each intra-cluster edge as $\sigma_{\text{intra}} = \min\{1.2\sigma, \sigma \cdot m/m_{\text{intra}}\}$, and the removing probability of each inter-cluster edge is decreased to $\sigma_{\text{inter}} = (\sigma \cdot m - \sigma_{\text{intra}} \cdot m_{\text{intra}})/m_{\text{inter}}$ to make sure that in expectation $\sigma \cdot m$ edges are removed as in the uniform strategy.

When conducting the contrastive learning procedure following GRACE [205], one augmentation branch used the uniformly random edge removing strategy, and the other branch adopted either the uniform or the clustered strategy. Both branches included a random feature masking with ratio 0.3. For these two GCL methods based on different augmentation strategies, the experiment setup is as follows: both methods used a GCN encoder with the same architecture and hyper-parameters (e.g., 2 convolutional layers with the embedding dimension of 32). Both performed 1000 training iterations to obtain node representations, whose quality was evaluated by using them as features for a downstream node classification task.

We compared the clustered augmentation based GCL with the uniform augmentation based GCL from two aspects: their performance in the downstream task and the spectral change on the augmented graphs. When evaluating the GCL performance, we randomly split the nodes into training, validation and test set with ratio $10\%, 10\%$ and 80% and a linear Logistic classifier was trained to conduct the task. The right-side of Figure 13 reports the mean and standard derivation of *F1 score* for 10 runs with different random seeds, which measures the downstream task performance. Meanwhile, we calculated the eigenvalues of the normalized Laplacian matrix of the input graph ($\mathbf{\Lambda}$), the augmented graphs with uniform strategy ($\mathbf{\Lambda}'_{\text{uniform}}$) and the augmented graphs with clustered strategy ($\mathbf{\Lambda}'_{\text{clustered}}$). The left-side of Figure 13 reports the L_2 distance of eigenvalues between the input and augmented graphs (e.g., $\|\mathbf{\Lambda} - \mathbf{\Lambda}'_{\text{uniform}}\|_2$ and $\|\mathbf{\Lambda} - \mathbf{\Lambda}'_{\text{clustered}}\|_2$) to measure the spectral change.

1.3 Summary of Datasets

The proposed GCL-TAGS is evaluated on 25 graph datasets. Specifically, for the *node classification task*, we included Cora, Citeseer, PubMed citation networks [152], Wiki-CS hyperlink network [119], Amazon-Computer and Amazon-Photo co-purchase network [153], and Coauthor-CS network [153]. For the *graph classification and regression tasks*, we employed TU biochemical and social networks [125], Open Graph Benchmark (OGB) [63] and ZINC [64, 48] chemical molecules, and Protein-Protein

Table 16: TU Benchmark Datasets [125] for graph classifcation task in unsupervised learning setting. The metric used for classification task is *accuracy*.

Data Type	Name	#Graphs	Avg. #Nodes	Avg. #Edges	#Classes
Biochemical Molecules	NCI1	4,110	29.87	32.30	2
	PROTEINS	1,113	39.06	72.82	2
	MUTAG	188	17.93	19.79	2
	DD	1,178	284.32	715.66	2
Social Networks	COLLAB	5,000	74.5	2457.78	3
	REDDIT-BINARY	2,000	429.6	497.75	2
	REDDIT-MULTI-5K	4,999	508.8	594.87	5
	IMDB-BINARY	1,000	19.8	96.53	2
	IMDB-MULTI	1,500	13.0	65.94	3

Interaction (PPI) biological networks [64, 207]. We summarize the statistics of these datasets and briefly introduce the experiment settings on them.

– A collection of datasets were used to evaluate *node classification* performance in both *unsupervised learning* and *adversarial attack* settings, and Table 15 summarizes the statistics of these datasets. **Cora, Citeseer, PubMed** citation networks [152] contain nodes representing documents and edges denoting citation links. The task is to predict the research topic of a document given its bag-of-word representation. **Wiki-CS** hyperlink network [119] consists of nodes corresponding to Computer Science articles, with edges based on hyperlinks. The task is to predict the branch of the field about the article using its 300-dimension pretrained GloVe word embeddings. **Amazon-Computer, Amazon-Photo** co-purchase networks [153] have nodes being items and edges representing that two items are frequently bought together. Given item reviews as bag-of-word node features, the task is to map items to their respective item category. **Coauthor-CS** network [153] contains node to be authors and edges to be co-author relationship. Given keywords of each author's papers, the task is to map authors to their respective field of study. All of these datasets are included in the PyG (PyTorch Geometric) library[12].

– Two sets of datasets were used to evaluate *graph prediction* tasks under the *unsupervised learning* setting. **TU Datasets** [125] provides a collection of benchmark datasets, and we used several biochemical molecules and social networks for graph classification as summarized in Table 16. The data collection is also included in the PyG library following a 10-fold evaluation data split. We used these datasets for evaluation of the graph classification task in unsupervised learning setting. **Open Graph Benchmark (OGB)** [63] contains datasets for chemical molecular property classification and regression tasks, which are summarized in Table 17. This data collection can be load via the OGB platform [13], and we used its processed format available in PyG library.

– A set of biological and chemical datasets were used to evaluate *graph classification* task under the *transfer learning* setting, summarized in Table 18. Following the transfer learning pipeline in [64], an encoder was first pre-trained on a large biological Protein-Protein Interaction (**PPI**) network or **ZINC** chemical molecule dataset, and then was evaluated on small datasets from the same domains.

[12] https://pytorch-geometric.readthedocs.io/en/latest/index.html
[13] https://ogb.stanford.edu/docs/graphprop/

Table 17: OGB chemical molecular datasets [63] for both graph classification and regression tasks in unsupervised learning setting. The evaluation metric used for regression task is *RMSE*, and for classification is *ROC-AUC*.

Task Type	Name	#Graph	Avg. #Nodes	Avg. #Edges	#Tasks
Regression	ogbg-molesol	1,128	13.3	13.7	1
	ogbg-molipo	4,200	27.0	29.5	1
	ogbg-molfreesolv	642	8.7	8.4	1
Classification	ogbg-molbace	1,513	34.1	36.9	1
	ogbg-molbbbp	2,039	24.1	26.0	1
	ogbg-molclintox	1,477	26.2	27.9	2
	ogbg-moltox21	7,831	18.6	19.3	12
	ogbg-molsider	1,427	33.6	35.4	27

Table 18: Biological interaction and chemical molecular datasets [64] for graph classification task in transfer learning setting. The evaluation metric is *ROC-AUC*.

Data Type	Stage	Name	#Graph	Avg. #Nodes	Avg. #Degree
Protein-Protein Interaction Networks	Pre-training	PPI-306K	306,925	39.82	729.62
	Fine-tuning	PPI	88,000	49.35	890.77
Chemical Molecules	Pre-training	ZINC-2M	2,000,000	26.62	57.72
	Fine-tuning	BBBP	2,039	24.06	51.90
		Tox21	7,831	18.57	38.58
		SIDER	1,427	33.64	70.71
		ClinTox	1,477	26.15	55.76
		BACE	1,513	34.08	73.71
		HIV	41,127	25.52	54.93
		MUV	93,087	24.23	52.55
		ToxCast	8,576	18.78	38.52

1.4 Model Analysis

• **Influence of Choosing Eigenvalues** To reduce the time complexity of eigen-decomposition when calculating the spectrum norm $\mathcal{L}_{\text{GS}}(\mathbf{\Delta})$, we can approximate the norm by only using the K lowest- and highest-eigenvalues. The time complexity of optimizing the augmentation scheme in Eq. Eq (15) with T iterations is $\mathcal{O}(TKn^2)$. This experiment shows the influence of K to the resulting GCL performance. Since the graphs encountered in the node prediction tasks are much larger than those in graph prediction tasks, we used the node classification datasets in Table 15 to conduct this experiment. Specifically, we test influence of K on four large graphs representing different types of networks: PubMed citation network, Wiki-CS hyperlink network, Amazon-Computers co-purchase network and Coauthor-CS network. We tuned K among $\{50, 100, 200, 500, 1000, 5000\}$ for each of the datasets containing $n \geq 10,000$ nodes. The other components of GCL maintained the same, except the resulting augmentation scheme using spectrum norm with different K.

Figure 25 demonstrates the performance of GCL-TAGS on the node classification task when different K was used to generate the augmentation scheme. The x-axis denotes the value of K with "all" indicating that all the eigenvalues were used. The performance decreases marginally when we used a smaller

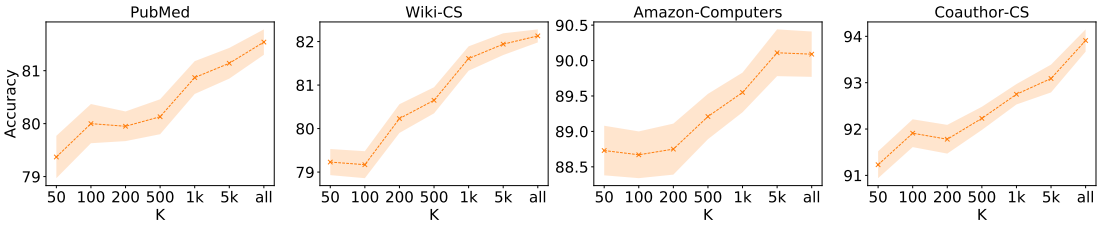


Fig. 25: Node classification performance when choosing K lowest- and highest-eigenvalues

Table 19: Node classification performance under *unsupervised* setting. We plug the spectrum based augmentation to different GCL frameworks, highlighted with suffix *-TAGS*. The metric is *accuracy%*. **Bold** highlights that the GCL framework with plugging our augmentation method outperforms its original version with p-value≤ 0.05.

Dataset	Cora	Citeseer	PubMed	Wiki-CS	Amazon-Computer	Amazon-Photo	Coauthor-CS
GRACE [205]	83.33±0.43	72.10±0.54	78.72±0.13	80.14±0.48	89.53±0.35	92.78±0.30	91.12±0.20
GRACE-TAGS	**84.21±0.51**	**72.87±0.58**	**79.94±0.22**	80.63±0.47	89.95±0.41	92.56±0.34	**91.98±0.20**
BGRL [169]	83.63±0.38	72.52±0.40	79.83±0.25	79.98±0.13	90.34±0.19	93.17±0.30	93.31±0.13
BGRL-TAGS	**84.34±0.42**	72.73±0.44	**80.78±0.32**	**81.04±0.22**	90.12±0.21	**93.58±0.39**	93.77±0.21
GBT [11]	80.24±0.42	69.39±0.56	78.29±0.43	77.30±0.62	88.02±0.32	92.23±0.35	92.85±0.31
GBT-TAGS	**82.43±0.51**	**71.12±0.48**	**80.05±0.49**	**78.89±0.54**	**89.04±0.43**	92.78±0.43	92.95±0.37
MVGRL [59]	85.16±0.52	72.14±1.35	80.13±0.84	77.52±0.08	87.52±0.11	91.74±0.07	92.11±0.12
MVGRL-TAGS	85.86±0.57	**72.76±0.63**	**81.54±0.24**	**82.13±0.15**	90.09±0.32	**93.52±0.26**	**93.91±0.24**

K, and generally when $K = 1000$ we can still achieve a comparable performance. This suggests that low and high eigenvalues are already quite informative in capturing graph structural properties. Similar phenomenon is also discussed in previous works [101]: small eigenvalues carry smoothly varying signals (e.g., similar neighbor nodes within the same cluster), while high eigenvalues carry sharply varying signals (e.g., dissimilar nodes from different clusters).

Gain of Spectrum Guided Augmentation on Other GCL Frameworks

In this experiment, we use an ablation study to evaluate the effectiveness of the graph spectrum guided topology augmentation scheme when applied to different contrastive learning frameworks. We focus on GCL for node-level representation learning, as this line of work adopts distinct contrastive objectives (e.g., bootstrapping in BGRL, and Barlow twins in GBT) and contrastive modes (e.g., node v.s. node in GRACE, and node v.s. graph in MVGRL), such that we can comprehensively demonstrate the effectiveness of our proposed augmentation in covering a variety of GCL frameworks.

Specifically, we replace the original uniformly random edge removing augmentation in GRACE, BGRL, GBT, and the diffusion matrix based augmentation in MVGRL with the proposed spectrum based augmentation scheme, and use *-TAGS* as suffix to denote them. Note that MVGRL-TAGS is basically GCL-TAGS since it uses the same contrastive objective as in MVGRL such that both node- and graph-level representations are obtained to serve a broader range of downstream tasks.

Table 19 shows the results of plugging our augmentation scheme on four types of GCL frameworks. We can observe that our augmentation scheme does not depend on a particular contrastive objective, but brings a clear performance gain across different GCL frameworks. Intuitively, our augmentation captures the essential structural properties by perturbing edges that cause large spectral change. Therefore, no matter what contrastive objective or mode is used, maximizing the correspondence of two views en-

courages the encoder to ignore the information carried by such sensitive edges. This demonstrates the importance of studying graph spectral properties for graph augmentation.

2 Additional Experimental Results in Chapter 3, Section 1

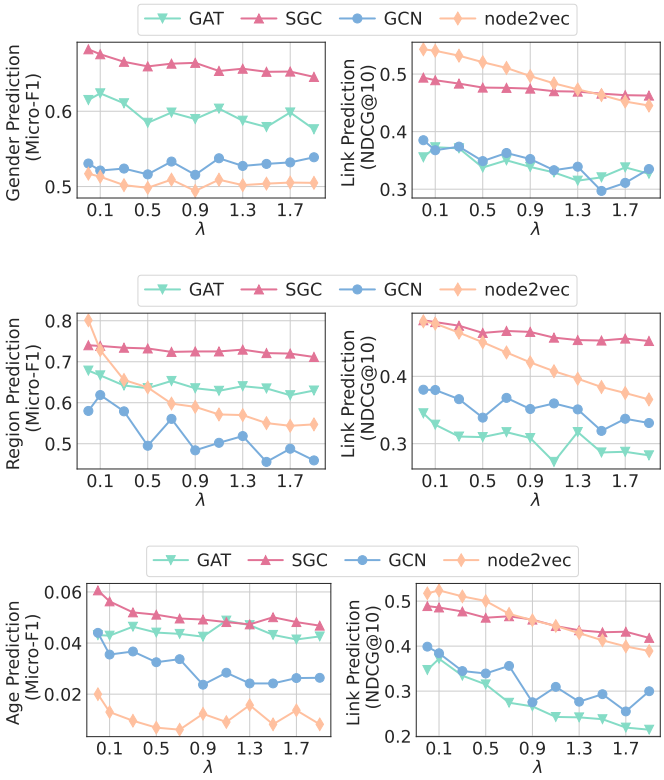


Fig. 26: Unbiasedness and utility trade-off using different regularization weights on UGE-C (x-axis). The left columns shows unbiasedness (attribute prediction), and the right columns shows utility (link prediction).

In Appendix 2.1, we include additional experiment results to report the trade-off between unbiasedness and utility on the complete set of embedding models on Pokec-z. In Appendix 2.2, we show a complete comparison among our proposed instances of unbiased graph embedding UGE-W, UGE-R and UGE-C. In Appendix 2.3, we investigate the influence of the regularization weight on the complete set of embedding models.

2.1 Additional Analysis on Undebiasedness

Table 20 summarizes the debiasing and utility performance of the proposed method and baselines when using four graph neural networks on Pokec-z. Each line of attribute prediction result is followed by

Table 20: The prediction performance of node embeddings learned on Pokec-z using four graph neural networks as embedding models. In each row, we use bold to mark the best debiasedness on attribute prediction or utility on link prediction.

Dataset	Embedding Model	Prediction Target	No Debiasing	Debiasing Method					Random
				Fairwalk	CFC	UGE-W	UGE-R	UGE-C	
Pokec-z	GAT	Gender (Micro-F1)	0.6232	0.6135	0.5840	0.6150	0.6094	**0.5747**	0.4921
		Link (NDCG@10)	0.3618	0.3280	0.2757	**0.3554**	0.3422	0.3376	0.0570
		Region (Micro-F1)	0.8197	0.8080	0.7217	0.6784	0.7660	**0.6356**	0.4966
		Link (NDCG@10)	0.3618	0.3287	0.2757	0.3451	**0.3547**	0.3098	0.0570
		Age (Micro-F1)	0.0526	0.0522	0.0498	0.0431	0.0545	**0.0429**	0.0007
		Link (NDCG@10)	0.3618	0.3122	0.2757	0.3471	0.3205	**0.3718**	0.0570
	SGC	Gender (Micro-F1)	0.6766	0.6631	**0.6520**	0.6822	0.6531	0.6596	0.4921
		Link (NDCG@10)	0.4975	0.4461	0.4011	**0.4938**	0.4850	0.4765	0.0570
		Region (Micro-F1)	0.7806	0.7820	**0.7150**	0.7402	0.7680	0.7323	0.4966
		Link (NDCG@10)	0.4975	0.4460	0.4011	**0.4832**	0.4799	0.4644	0.0570
		Age (Micro-F1)	0.0621	0.0662	0.0654	0.0606	0.0529	**0.0510**	0.0007
		Link (NDCG@10)	0.4975	0.4461	0.4011	**0.4889**	0.4694	0.4630	0.0570
	GCN	Gender (Micro-F1)	0.5532	0.5589	0.5493	0.5306	0.5301	**0.5162**	0.4921
		Link (NDCG@10)	0.3865	0.2807	0.3836	**0.3851**	0.3727	0.3488	0.0570
		Region (Micro-F1)	0.7445	0.7616	0.7693	0.5800	0.6105	**0.4951**	0.4966
		Link (NDCG@10)	0.3865	0.2807	**0.3836**	0.3801	0.3360	0.3386	0.0570
		Age (Micro-F1)	0.0425	0.0416	0.0391	0.0439	0.0409	**0.0324**	0.0007
		Link (NDCG@10)	0.3865	0.2807	0.3836	**0.3987**	0.3550	0.3391	0.0570
	node2vec	Gender (Micro-F1)	0.5248	0.5347	0.5137	0.5171	**0.4949**	0.4982	0.4921
		Link (NDCG@10)	0.5491	0.5120	**0.5496**	0.5430	0.5463	0.5206	0.0570
		Region (Micro-F1)	0.8423	0.8462	0.8423	0.8012	0.6490	**0.6372**	0.4966
		Link (NDCG@10)	0.5491	0.5120	**0.5496**	0.4816	0.5354	0.4506	0.0570
		Age (Micro-F1)	0.0365	0.0404	0.0365	0.0200	0.0122	**0.0068**	0.0007
		Link (NDCG@10)	0.5491	0.5120	**0.5496**	0.5173	0.5439	0.5002	0.0570

the corresponding performance on link prediction. Generally, UGE-W achieves the best link prediction performance and UGE-R has better debiasing effect. Combining UGE-W with UGE-R produces UGE-C with better trade-off.

2.2 Ablation Study

Figure 27 presents the performance of three proposed model (UGE-W, UGE-R and UGE-C) applied to four graph neural networks (GAT, SGC, GCN and node2vec). We can clearly observe that in most cases UGE-R has better debiasing effect compared with UGE-W, while UGE-W can better maintain the utility for downstream link prediction task. UGE-C as the combination of them indeed makes the best of the both designs.

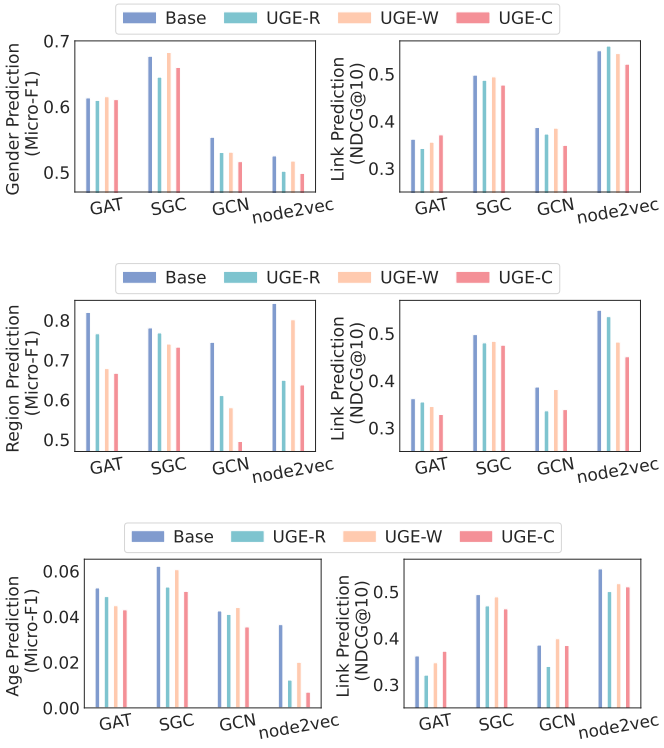


Fig. 27: Comparison among our proposed models on different embedding models. The left columns shows the unbiasedness (attribute prediction) and the right columns shows the utility (link prediction).

2.3 Unbiasedness-Utility Tradeoff in UGE

We now include a complete analysis on unbiasedness and utility trade-off across embedding models in Figure 26. It clearly shows a trade-off: as the weight increases, we obtain a stronger debiasing effect with a cost of the utility on link prediction.

3 Proofs and Detailed Attack Objectives in Section 2

We list the detailed gradient calculation of the spectral distance term with eigen-decomposition, the proof of Theorem 1, and the attack objectives for different white-box variants of SPAC in Section 2.

3.1 Gradient of the Spectral Distance

Recall that we obtain the following form via the chain rule:

$$\frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \mathbf{\Delta}_{ij}} = \sum_{k=1}^{n} \frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \lambda'_k} \sum_{p=1}^{n} \sum_{q=1}^{n} \frac{\partial \lambda'_k}{\partial \mathbf{L}'_{pq}} \frac{\partial \mathbf{L}'_{pq}}{\partial \mathbf{\Delta}_{ij}}$$

Here is the detailed calculation of each component:

$$\frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \lambda'_k} = \frac{\lambda'_k - \lambda_k}{\|\mathbf{\Lambda} - \mathbf{\Lambda}'\|_2}, \frac{\partial \lambda'_k}{\partial \mathbf{L}'_{pq}} = \mathbf{u}'_{kp}\mathbf{u}'_{kq}, \frac{\partial \mathbf{L}'_{pq}}{\partial \mathbf{\Delta}_{ij}} = \frac{\mathbf{C}_{ij}}{2\sqrt{d'_p d'_q}}(\mathbb{1}_{i=p}\frac{\mathbf{A}'_{pq}}{d'_p} + \mathbb{1}_{j=q}\frac{\mathbf{A}'_{pq}}{d'_p} - 2\mathbb{1}_{i=p,j=q})$$

where d'_p is the degree on node p of the perturbed graph: $d'_p = \sum_{k=1}^{n} \mathbf{A}'_{kp}$, and similarly $d'_q = \sum_{k=1}^{n} \mathbf{A}'_{kq}$. Meanwhile, $\mathbf{A}'$ is the adjacency matrix of the perturbed graph. The indication function $\mathbb{1}_{\text{condition}}$ is 1 if the condition is true, otherwise it is 0.

3.2 Proof of Theorem 1

Proof. **Theorem 1.** For the generalized eigenvalue problem: $\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{M}\mathbf{u}_i$, if the matrix is slightly perturbed $\mathbf{L}' = \mathbf{L} + \nabla\mathbf{L}$, we aim to find the corresponding eigenvalue perturbation: $\lambda'_i = \lambda_i + \nabla\lambda_i$. From eigenvalue perturbation theory [157], we have

$$\lambda'_{\mathbf{i}} - \lambda_{\mathbf{i}} \approx \mathbf{u}_{\mathbf{i}}^{\top}(\nabla\mathbf{L} - \lambda_{\mathbf{i}}\nabla\mathbf{M})\mathbf{u}_{\mathbf{i}}$$

And for a normalized graph Laplacian $\mathbf{L}' = \mathbf{L} + \nabla\mathbf{L}$, we have $\nabla\mathbf{M} = \text{diag}(\nabla\mathbf{L} \cdot \mathbf{1}_n)$. Submitting $\nabla\mathbf{M}$ concludes the proof.

3.3 Attack Objectives for White-box Variants

Recall that SPAC can be flexibly combined with the white-box attack framework as shown in Eq (47), which consists of a task-specific attack objective $\mathcal{L}_{\text{attack}}$ and the proposed SPAC objective $\mathcal{L}_{\text{SPAC}}$. We denote the training node set as V_0 and test node set as V_t. Different choices of $\mathcal{L}_{\text{attack}}$ result in the following variants.

SPAC-CE combines SPAC with PGD-CE [193], and maximizes the *cross-entropy* loss on the target *test set* for *evasion* attack:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_t} \text{crossEntropy}(f_\theta(\mathbf{A} + \mathbf{\Delta}, \mathbf{X})_i, y_i)$$

SPAC-C&W combines SPAC wih PGD-C&W [193], and maximizes the *negative* C&W *score* on the target test set for *evasion* attack:

$$\mathcal{L}_{\text{attack}} = -\sum_{v_i \in V_t} \max\{Z_{i,y_i} - \max_{c \neq y_i} Z_{i,c} - \kappa\}$$

where $Z_{i,c}$ denotes the prediction logit on label c, and $\kappa \geq 0$ is a confidence level of making wrong decisions. Intuitively, the C&W score evaluates how good the model can differentiate the prediction on the ground-truth label and on the label with the (second) highest likelihood. So the attack aims to confuse the model by maximizing the negative C&W score.

SPAC-Min combines SPAC and Max-Min [193], and maximizes the *cross-entropy* loss on the *training set*, while a surrogate model $f_{\theta'}$ is iteratively retrained. The perturbed graph is then used to train a victim model, and we report the classification performance of the test set on clean graph. The $poisoned$ graph is generated by:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_0} \text{crossEntropy}(f_{\theta'}(\mathbf{A} + \mathbf{\Delta}, \mathbf{X})_i, y_i)$$

SPAC-Train combines SPAC with Meta-Train [209], and maximizes the *cross-entropy* loss on labeled *training nodes*, arguing that if a model has a high training error, it is likely to generalize poorly:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_0} \text{crossEntropy}(f_{\theta'}(\mathbf{A} + \mathbf{\Delta}, \mathbf{X})_i, y_i)$$

The objective is similar to SPAC-Min, but instead of retraining the surrogate model, SPAC-Train calculate *meta-gradients* on the perturbation matrix through the surrogate model.

SPAC-Self combines SPAC with Meta-Self [209], and maximizes the *cross-entropy* loss on unlabeled *test nodes* which are assigned pseudo labels predicted by the model trained on tbe clean graph:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_t} \text{crossEntropy}(f_{\theta'}(\mathbf{A} + \mathbf{\Delta}, \mathbf{X})_i, \widehat{y}_i)$$

where $\widehat{y}_i$ is the predicted label from the model trained on the clean graph f_θ.