12-11-2023

Towards Data Privacy and Utility in the Applications of Graph Neural Networks

Kainan Zhang

Recommended Citation
Zhang, Kainan, "Towards Data Privacy and Utility in the Applications of Graph Neural Networks." Dissertation, Georgia State University, 2023.
doi: https://doi.org/10.57709/36369498

Towards Data Privacy and Utility in the Applications of Graph Neural Networks

by

Kainan Zhang

Under the Direction of Zhipeng Cai, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2023

ABSTRACT

Graph Neural Networks (GNNs) are essential for handling graph-structured data, often containing sensitive information. It's vital to maintain a balance between data privacy and usability. To address this, this dissertation introduces three studies aimed at enhancing privacy and utility in GNN applications, particularly in node classification, link prediction, and graph classification. The first work tackles celebrity privacy in social networks. We develop a novel framework using adversarial learning for link-privacy preserved graph embedding, which effectively safeguards sensitive links without compromising the graph's structure and node attributes. This approach is validated using real social network data. In the second work, we confront challenges in federated graph learning with non-independent and identically distributed (non-IID) data. We introduce PPFL-GNN, a privacy-preserving federated graph neural network framework that mitigates overfitting on the client side and inefficient aggregation on the server side. It leverages local graph data for embeddings and employs embedding alignment techniques for enhanced privacy, addressing the hurdles in federated learning on non-IID graph data. The third work explores Few-Shot graph classification, which aims to classify novel graph types with limited labeled data. We propose a unique framework combining Meta-learning and contrastive learning to better utilize graph structures in molecular and social network datasets. Additionally, we offer benchmark graph datasets with extensive node-attribute dimensions for future research. These studies collectively advance the field of graph-based machine learning by addressing critical issues of data privacy and utility in GNN applications.

INDEX WORDS: Machine Learning, Graph Neural Networks, Privacy Preservation

Towards Data Privacy and Utility in the Applications of Graph Neural Networks

by

Kainan Zhang

Committee Chair:	Zhipeng Cai
Committee:	Zhipeng Cai
	Yingshu Li
	Wei Li
	Yan Huang

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2023

DEDICATION

I dedicate this dissertation to my family, especially my parents, Zhengyu Zhang and Zhongying Li. Their unwavering encouragement and heartfelt words have been my guiding light throughout this journey.

I also extend my gratitude to my friends and colleagues. Their steadfast support and companionship during this doctoral program have been invaluable. Their contributions, in myriad ways, have enriched this endeavor, and I'm eternally thankful.

Embarking on this Ph.D. journey introduced me to the complexities and nuances of computer science. Beyond algorithms and codes, I learned about resilience, perseverance, and the insatiable thirst for knowledge. As I pause to reflect on this significant chapter, I'm reminded that in the vast canvas of computer science, every byte of data has an untold story, and every line of code is a legacy in the making.

ACKNOWLEDGMENTS

I wish to thank my committee members for their invaluable insights and dedication. I'm particularly indebted to Dr. Zhipeng Cai, my committee chairman, for his unwavering support, constructive feedback, encouragement, and above all, his enduring patience throughout this journey. My sincere appreciation goes out to Dr. Yingshu Li, Dr. Wei Li, and Dr. Yan Huang for their willingness and commitment to being part of my committee.

I'm also thankful to my school division for granting me the opportunity to conduct my research and for their continuous assistance. A special nod of appreciation to the esteemed professors and faculty of the computer science department, whose guidance and encouragement have been instrumental in this academic endeavor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background of Graph Learning

Graphs are a powerful and versatile tool for modeling complex systems and relationships, which can represent a wide variety of data types, including social networks, biological networks, and physical systems. Graph learning, also known as graph-based machine learning, is a subfield of machine learning that deals with the analysis and prediction of graph-structured data, which has its roots in graph theory and combinatorial optimization. Although these topics have been studied for many decades in mathematics and computer science, there has been a growing interest in applying these techniques to problems with the increasing availability of large-scale graph-structured data and advances in machine learning.

Recent developments in graph representation learning and graph neural networks (GNNs) have significantly contributed to the advancement of graph learning. Representation learning methods aim to learn a low-dimensional embedding of nodes in a graph that captures the underlying structure and properties of the graph. GNNs, on the other hand, are neural network architectures designed to operate on graph-structured data in various tasks such as node classification, link prediction, and graph classification.

In conclusion, graph learning is a rapidly growing field that has the potential to enable the development of intelligent systems that can analyze and reason complex graph-structured data. The field continues to evolve with new advancements in graph theory, machine learning, and computer science, making it an exciting area of research.

1.2 Privacy Issues in Graph Neural Networks

Graph Neural Networks (GNNs), known for their effectiveness in tasks like node classification and link prediction, raise notable privacy concerns [1]. Among these concerns is the risk of inferring sensitive information about individual nodes or edges within a graph, even when such information is not explicitly provided as input to the model, resembling the inference attack [2, 3]. In the context of autonomous driving systems, both graph data and image data introduce unique privacy challenges. Graph data, employed in systems analyzing traffic patterns and road infrastructure, presents significant concerns regarding privacy leakage. For instance, an inference attack within a Graph Neural Network (GNN) could potentially unveil sensitive information about a specific vehicle, like its location history or driving behavior, by discerning patterns within publicly available traffic data. This privacy issue arises because GNNs inherently encode personal data into their structural representations, making users susceptible to privacy breaches and linkage attacks. Conversely, image data used in autonomous vehicles, encompassing camera footage of road scenes and nearby objects, introduces its own privacy challenges [4, 5]. Although the raw image data may not overtly disclose personal information, privacy concerns may arise when images inadvertently capture identifiable details such as license plates, faces, or unique vehicle features [6]. These concerns relate more to the potential for privacy breaches through unintentional data collection and storage.

These privacy challenges contrast with those encountered in image and video recognition [7], as GNNs inherently encode personal data into their structural representations, thus

exposing users to privacy breaches and linkage attacks. In contrast, image recognition primarily deals with anonymized pixel-level data [6], presenting distinct yet equally important privacy challenges. Both domains stand to gain from privacy-preserving techniques like differential privacy [8], though their application complexities vary, and they share the intricate dynamics of managing data owned by multiple parties.

To address these privacy concerns, researchers [9] are exploring methods for privacy-preserving GNNs. These methods include differential privacy [10], which adds noise to the inputs or outputs of a model to prevent the inference of sensitive information, and homomorphic encryption, which allows computation on encrypted data without revealing the underlying plain text. In addition to these methods, there are recent works in the field of federated learning, which allow multiple parties to train a model on their data while keeping it private. While these techniques can help protect privacy, they may suffer from reduced model performance or increased computational complexity. Therefore, it is critical to consider the trade-offs between data privacy protection and model utility.

1.3 Federated Graph Learning on Non-IID Graph Data

Federated learning (FL) is a learning strategy for training the model on decentralized data. In federated learning, the goal is to train a global model with better scalability while preserving local clients' data privacy.

One of the challenges in federated graph learning is dealing with non-IID (non-identically and independently distributed) graph data. It usually happens because the graph data of

different clients may have different properties or structures when collected under different conditions.

To address this privacy issue and enhance the performance of federated graph learning, several strategies can be employed. One approach is to leverage domain adaptation techniques that aim to align the distributions of data across different client nodes, thereby mitigating domain shift issues. Techniques like adversarial training can be employed to learn a shared feature space for all clients, promoting alignment and reducing disparities among data distributions. Transfer learning is another avenue to explore, allowing knowledge from one node to improve the performance of another. Additionally, research into embedding alignment methods is promising, focusing on aligning the node representations in a graph across various clients, ultimately enhancing both privacy and model effectiveness.

1.4 Few-Shot Graph Learning

Few-shot graph learning focuses on developing algorithms that can learn from a small number of samples or "shots" of graph-structured data. The goal of few-shot graph learning is to enable the model to generalize to new graphs or new classes of nodes or edges based on the limited number of training data.

The main challenge of few-shot graph learning is learning a generalizable graph representation from a small number of training samples. To overcome this challenge, researchers have proposed various methods: (i) Meta-learning algorithms aim to learn the generalizable knowledge from the few-shot examples, which can be used to adapt to new tasks quickly; (ii)

Transfer learning leverages knowledge from one task or domain to improve performance on another task or domain [11]; (iii) Graph similarity methods aim to find similar graphs between the training set and the test graph, then use the knowledge from these similar graphs to make predictions on the test graph.

In this dissertation, we focus on research on the above topics. By investigating practical problems in each domain, we propose effective methods that can provide better data privacy protection and usability in the applications of GNNs, respectively.

First, aiming at the privacy protection of graph learning, we propose a link-privacy preserved graph embedding framework using adversarial learning, which can reduce the adversary's prediction accuracy on sensitive links while persevering sufficient non-sensitive information such as graph topology and node attributes in graph embedding.

Second, we design a federated graph learning framework combined with the embedding alignment technique. Because the server only needs to integrate client-preferred public information, it can significantly reduce the risk of privacy disclosure during the learning process. The embedding alignment technique ensures that the clients holding non-IID data can change information. Furthermore, we find that injecting aligned information into the local model has regularization effects empirically and thus avoids model overfitting.

Third, by investigating a general scenario for few-shot graph classification tasks, we propose a learning framework that integrates both Meta-learning and contrastive learning techniques into an end-to-end process to obtain accurate graph classification results. We also construct two general multi-class benchmark graph datasets with large node-attribute

dimensions to facilitate future research on few-shot graph classification.

We organized the dissertation as follows: Related works on the above topics in graph learning are introduced in Chapter 2. Then, we present the details of each research in Chapter 3, Chapter 4, and Chapter 5, respectively. Finally, we make a conclusion about this dissertation in Chapter 6.

CHAPTER 2
RELATED WORKS

2.1 Graph Neural Networks

2.1.1 Graph Embedding Methods

DeepWalk [12] marks a significant step in unsupervised graph representation learning by using random walks across a network and applying the skip-gram technique [13] to identify underlying vertex patterns. Following the same architectural principle, Node2Vec [14] refines vertex sampling using an enhanced random walk strategy, effectively capturing the graph's structural similarities and connections. Meanwhile, LINE [15] emphasizes the preservation of both local and global network features through two specific objectives—first-order and second-order proximities. This approach proves effective for networks with directionality or weight attributes. Building on LINE's foundation, SDNE [16] employs an autoencoder to concurrently optimize the two proximity objectives, offering a solution especially effective for networks with sparse connections. Moving beyond merely focusing on graph structures, GAE [17] harnesses graph convolutional networks (GCNs) to encode node information, enhancing its capability by factoring in node features. Subsequent studies [18, 19] validate GAE's efficacy, especially in tasks like link prediction and recommendation systems. While substantial progress has been made in graph embedding techniques, the aspect of privacy in embedding outputs has received insufficient attention, akin to the growing concern surrounding the popularity of backdoor attacks in image models, highlighting significant security and privacy risks in both graph-based and image-based machine learning applications.

2.1.2 Embedding Space Alignment

Embedding techniques have gained significant traction in the realms of machine learning and graph analysis [20]. In this context, aligning various embedding spaces is crucial, drawing parallels to language translation in bridging communication gaps between different languages. Leading the charge in alignment methodologies, cross-lingual word embedding alignments have seen a surge in interest and development over recent years [21]. Notable frameworks like MUSE [22] and VecMap [23] offer cutting-edge toolkits tailored for Bilingual Lexical Induction (BLI) datasets. As applications rooted in knowledge extraction such as question answering and knowledge graph completion evolve, there's been a marked increase in research focusing on knowledge graph embedding alignments [24, 25]. These comprehensive investigations underscore the potential of harnessing alignment techniques not merely as training objectives but as pivotal instruments for information extraction and integration throughout the training phase.

2.1.3 Graph Contrastive Learning

Graph Contrastive Learning (GCL) aptly derives its name from its primary function: contrasting graph samples. It ensures that samples with similar distributions are drawn closer in the embedding space, while those from distinct distributions are pushed apart. With contrastive learning [26] serving as the foundation of GCL, recent studies have delved into the intricacies of graph augmentation strategies. GraphCL [27] examines four distinct graph augmentations, integrating various priors for the unsupervised representation of graph data.

Building on this, the authors [28] introduce JOint Augmentation Optimization, an encompassing bi-level optimization structure that automates data augmentations. GCA [29], in a similar vein, offers an adaptive augmentation technique that takes into account the topological and semantic nuances of a graph.

Several initiatives aim to generate enriched graph views. GRACE [30] devises a dual-layered approach, amalgamating structure, and attribute levels to ensure a variety of node contexts. InfoGCL [31] adopts the Information Bottleneck principle, aiming to trim down the mutual information among contrastive segments. Meanwhile, AD-GCL [32] sidesteps redundancy by honing adversarial graph augmentation strategies.

Sampling bias presents a recurring challenge in GCL. To counter this, Lin et al. [33] champion a prototype-centric clustering method, and Yu et al. [34] forgo graph augmentations, opting instead to infuse uniform noises to bolster the evenness of the resulting representations. Recognizing the escalating interest in GCL, PyGCL [35] emerges as a bench-marking tool, furnishing empirical insights into prevailing GCL algorithms and illuminating pathways for future exploration.

GCL can be also practically applied in blockchain-based IoT systems to enhance security, privacy, and insights. For instance, in a large-scale IoT network utilizing blockchain, this technique can be employed to learn meaningful representations of IoT devices, transactions, and interactions [36, 37]. These representations aid in device identification, anomaly detection, and tracking within the network, bolstering security and fraud detection [38]. Graph representations learned through contrastive learning also enable network visualiza-

tion, helping administrators understand network topology and data flows intuitively. Furthermore, this approach allows for privacy-preserving analysis while facilitating knowledge transfer between similar IoT networks, ultimately improving the efficiency and effectiveness of blockchain-based IoT applications.

2.2 Link Privacy Preservation in Social Networks

The deepening exploration of Online Social Networks (OSNs) has magnified concerns surrounding user privacy. Korolova et al. [39] demonstrate how attackers, with minimal information from a few users, can reconstruct the link structure of an entire network. Ying and Wu [40] assess the efficacy of edge randomization in safeguarding sensitive link privacy and highlight potential risks—attackers can improve link prediction accuracy using node proximity measures. In another study, Fire et al. [41] show that by utilizing minimal training data, one can reconstruct links removed for privacy reasons using a link prediction classifier.

Fard and Wang [42] introduce a structure-sensitive randomization approach that conceals sensitive links in directed graphs with minimal data distortion. Acknowledging the intricate relationship between utility/public attributes, private/public attributes, and link data, Cai et al. [43] fashion a comprehensive strategy to cleanse social networks, protecting against inference attacks on user profiles and their interrelations.

While these studies [44, 45] present a balance between data utility and privacy, they overlook the potential of integrating their approaches with graph embedding—a potent tool in graph analysis. Relevantly, DPNE [46] emerges as a pioneering effort to ensure differential

privacy in network embedding. PPGD [47] further introduces a differentially private gradient descent technique tailored for matrix factorization (MF)-based graph embedding matrix sharing. However, it's worth noting that even though differential privacy is often viewed as a pinnacle in privacy standards, it isn't foolproof against all privacy threats. The performances of both DPNE and PPGD in graph representation are, additionally, constrained by their reliance on matrix factorization methods.

2.3 Federated Learning

2.3.1 Federated Learning with Non-IID Dataset

Non-IID local data introduces statistical challenges to federated learning, impacting training convergence and markedly reducing accuracy. Addressing this, Zhao et al. [48] present a strategy to enhance the training of non-IID data by establishing a globally shared data fraction across all edge devices. Wang et al. [49], recognizing the bias induced by non-IID data, introduce Favor, an experience-driven control framework. This framework intelligently selects client devices for each federated learning round to balance the bias and hasten convergence.

Several federated learning algorithms aim to optimize learning efficiency in non-IID data scenarios. FedProx [50] emerges as a re-parametrized iteration of FedAvg, breaking ground in managing federated network heterogeneity. FedPD [51] investigates the non-convex behavior of the FedAvg algorithm, leading to a federated learning framework that boasts optimal rates and adaptivity to non-IID data. Similarly, Li et al. [52] unveil FedBN, incorporating local

batch normalization to counteract feature shifts prior to model averaging, thereby expediting convergence rates.

Despite these advances, Li et al. [53] demonstrate that no single state-of-the-art FL algorithm consistently outperforms others across diverse data partitioning strategies, which encompass typical non-IID data scenarios. For preserving differential privacy in non-IID federated learning contexts, Xiong et al. [54] craft the 2DP-FL algorithm, which employs adaptable noise to align with varied privacy benchmarks. However, while these methods make strides across various domains, they overlook the potential of leveraging graphs that inherently display non-IID characteristics as experimental datasets.

To further optimize federated graph learning, game theory can also be integrated into the framework [55]. Game-theoretical models can help analyze and design incentive mechanisms that encourage clients to actively participate while preserving their data privacy. By aligning the interests of clients and the global model, game theory can contribute to more efficient and privacy-conscious federated graph learning systems.

2.3.2 Federated Learning on Graph Neural Networks

In contrast to the extensive advancements in the vision and language domains, research on federated learning in graphs remains somewhat underrepresented. For instance, SGNN [56] employs a similarity-based graph neural network to grasp the structural information of nodes. Still, it primarily adopts the concept of federated learning to conceal original data from varying sources. In a similar vein, Lalitha et al. [57] introduce a distributed learning algorithm where nodes refine their beliefs through information aggregation from

neighbors, aiming to understand the optimal model for the entire network. The emergence of FedGraphNN [58] propels GNN-based federated learning research, positioning it as a pivotal federated learning system and benchmark. Yet, their findings underscore significant hurdles in federated GNN training. Notably, federated GNNs often underperform in datasets with a non-IID split compared to centralized GNNs, highlighting the need for intensified research in this sector.

Furthermore, federated GNN encounters fundamental challenges inherent to traditional federated settings, such as Expensive Communication [59], Systems Heterogeneity, Statistical Heterogeneity [60], and Privacy Concerns [50]. In response to the statistical heterogeneity of data, He et al. [61] unveil SpreadGNN, an innovative multi-task federated training framework. This approach operates even with partial labels and without a central server, leveraging Decentralized Periodic Averaging SGD for decentralized multi-task learning issues. Addressing privacy considerations, Sajadmanesh et al. [62] craft an architecture-agnostic GNN learning algorithm that boasts formal privacy assurances rooted in Local Differential Privacy. This mechanism also integrates features from multi-hop nodes to clarify noisy labels. Beyond foundational models and theoretical insights, the practical application of federated GNN warrants exploration. As a case in point, FedGNN [63] presents a federated structure for the GNN-centric recommendation system. This model facilitates the collective training of GNN models from dispersed user data while employing elevated user-item interaction data to fortify privacy. In this article's subsequent sections, we delve into these four challenges within our work and elaborate on the framework's versatility.

2.4 Few-Shot Learning

2.4.1 Few-Shot Learning and Meta-learning

Wang et al. [64] notably define FSL as a machine learning problem characterized by $\mathcal{E}$, $\mathcal{T}$, and $\mathcal{P}$. In this context, the machine draws from experience $\mathcal{E}$ with limited supervised data to address task $\mathcal{T}$, aiming to enhance the performance measure $\mathcal{P}$. Meta-learning, often termed "learning-to-learn", emerges as the prevailing framework for FSL in contemporary research [65]. This approach is perceived to offer a vital advantage: it identifies the coherence between training and testing objectives, equipping the model to derive insights directly from few-shot classification tasks [66]. The Matching Network [67] and the Prototypical Network [68] are two meta-learners that both incorporate a memory component via neural networks. While the Matching Network crafts shared representations for labeled examples and associate a new test instance with stored examples using cosine similarity, the Prototypical Network establishes a prototype vector space for individual classes. It then links the test instance to the prototype by determining the softmax likelihood based on a distance metric.

2.4.2 Few-Shot Learning on Graph Classification

Model-Agnostic Meta-learning (MAML) [69] is designed to identify an optimal model parameter initialization, enabling efficient generalization to new tasks using just a few gradient steps and a limited dataset. Expanding on this concept, Ma et al. [70] introduce a graph meta-learner that leverages GNN-based modules for swift adaptation to graph data. Additionally, they implement a step controller to ensure the robustness and generalizability of

the meta-learner. Taking cues from the graph's normalized Laplacian spectrum, Chauhan et al. [71] suggest a few-shot graph classification approach. This method taps into latent inter-class relationships constructed by a super-graph, where the L_p Wasserstein distance functions as the metric, clustering the super graphs to prototype graphs. On another front, SMF-GIN [72] offers a metric-based meta-learning framework for few-shot graph classification tasks. This framework, anchored on the Graph Isomorphism Network [73], meticulously incorporates both global and local structures of the input graph via an attention mechanism. Recently, Hassani et al. [74] put forth an attention-focused graph encoder. This innovative approach employs three coherent graph views to glean task-specific representations for rapid adaptation and task-agnostic information for streamlined knowledge transfer.

Meta-learning proves highly practical for enhancing the effectiveness of Internet of Things (IoT) applications, particularly those involving graph data, by providing heightened adaptability and operational efficiency. In IoT networks characterized by intricate graph structures depicting sensor relationships and data flows, meta-learning empowers models to swiftly adjust to shifts in sensor deployments, significantly improving system responsiveness. It plays a pivotal role in anomaly detection, dynamically adapting to evolving network topologies, and optimizing resource allocation, even in resource-constrained environments, thereby bolstering overall system performance. Furthermore, meta-learning facilitates seamless integration of heterogeneous data sources, streamlining knowledge transfer between different IoT deployments [75]. Additionally, it effectively manages the challenges posed by ever-changing network conditions. Moreover, meta-learning's capacity to enhance adaptability in edge

computing scenarios [76], enabling local data processing on IoT devices, solidifies its status as a valuable tool for addressing the multifaceted complexities of IoT applications [77].

CHAPTER 3

Link-privacy Preserving Graph Embedding Data Publication With Adversarial Learning

3.1 Introduction

In recent years, social networks have transitioned from mere entertainment platforms to being deeply interwoven into daily lives. As of January 2020, social media users numbered around 3.80 billion, growing at a rate of 7 percent annually. These networks not only facilitate interactions between users and their connections but also empower third parties to leverage social network data for diverse purposes [78]. These purposes range from business promotions to healthcare enhancements and even disaster prevention, all made possible through advanced data mining and machine learning techniques [79, 80, 81, 82, 83].

However, the intricate combinatorial structures of graph data hinder the broad application of many machine learning methods, which primarily accept vector representations. Graph embedding techniques offer a compelling solution, converting graph data into low-dimensional vectors, thus enabling the application of a broader array of statistical and machine learning tools [84]. It's imperative that these vectors retain not only the graph topology but also other pertinent graph information. Consequently, the release of graph embedding data poses a similar challenge as publishing raw social network data: the significant risk to user privacy. While it's possible to sanitize a social network by subtly altering the original graph to maintain utility and safeguard user privacy [85, 86], using graph embedding immediately after traditional data anonymization might undermine model effectiveness. Given

that dominant graph embedding techniques focus on both local neighborhood and global graph structures, revealing details about a removed node/user becomes plausible through the embedding vectors of adjacent nodes [87].

Recently, the concept of differential privacy entered the domain of graph embedding. This approach ensures the output of the disclosed graph embedding matrix remains statistically consistent, even when an edge in the original graph gets added or removed. However, the multi-dimensional nature of graph data makes it highly sensitive, possibly resulting in diminished utility and heightened computational costs. As a result, striking a balance between sharing social graph embeddings and maintaining user privacy becomes essential.

This work delves into the concept of "celebrity privacy," which alludes to the rights of celebrities and public figures to control information disclosure [88]. Unlike the general populace, celebrity privacy often faces scrutiny from the media for profit-driven motives or from fans due to personal interest. Furthermore, celebrities, given their expansive social networks, are more vulnerable to inference attacks [89]. A prime example would be deducing a relationship between Chinese basketball player Yao Ming and English footballer David Beckham, simply from their joint participation in a Wild-Aid campaign championed by Britain's Prince William [90]. This issue can be broadened to the challenge of preserving relationship data among users possessing numerous strong connections within a social network graph. Even though deleting a relationship link might obfuscate direct connections, higher-order data (such as mutual friends and attributes) ensures that the embeddings of these "sensitive" users remain closely aligned, allowing adversaries to predict relationships with considerable

accuracy [20]. Another challenge is privacy customization, wherein celebrities or data owners may wish to shield only specific relationship links.

To address these concerns and minimize privacy exposure risks in social network data release, we introduce the Link Privacy Preserved Graph Embedding (LPPGE) framework. This framework, rooted in adversarial learning, incorporates two preprocessing methodologies. The key takeaways of our research include:

- We investigate a practical privacy issue named celebrity privacy based on some graph embedding methods that are widely used for social network analysis, and our proposed work can protect the specified links upon user requirements.
- Our framework integrates both graph embedding and social network privacy protection into an end-to-end process flow through an adversarial training-based graph autoencoder.
- Extensive experiments on ground truth social network data demonstrate the performance of our framework in privacy protection compared with other existing methods.

3.2 Problem statement

We first introduce the definitions to state our problem as follows:

Definition 1. **Social network:** We model a social network as an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ consisting of a set of users $\mathbf{V}$, friendship-link set $\mathbf{E}$, and user-attribute set $\mathbf{X}$. $\mathbf{A}$ is the adjacency matrix corresponding to the structure of graph $\mathbf{G}$. If $\mathbf{e}(i, j) \in \mathbf{E}$ (*i.e.*, users u_i and u_j are friends), then $\mathbf{A}_{ij} = 1$, otherwise $\mathbf{A}_{ij} = 0$.

As we are interested in protecting the celebrity users, the sensitive user is defined as follows.

Definition 2. **Sensitive user:** Given a social network $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ and λ, if user $u_i \in \mathbf{V}$ and the node degree of $u_i \geq \lambda$, u_i is a sensitive user. The value of λ is pre-defined by the data owner, which is usually larger than the average degree of graph $\mathbf{G}$.

Definition 3. **Sensitive link:** Any pair of sensitive users u_i and u_j in social network $\mathbf{G}$ where $\mathbf{e}(i, j) \in \mathbf{E}$ is a candidate sensitive link. The candidate-sensitive link set is defined as CS. The data owner can customize the set of actual sensitive links $S \subseteq CS$. Links in $E \setminus CS$ are considered as non-sensitive links.

Graph embedding is generally used for a variety of machine-learning tasks, such as node classification and link prediction. The ultimate goal of our method is to publish link-privacy preserved graph embedding vectors without sacrificing data utility/usability. In the following, we define data utility and link privacy in our LPPGE.

Definition 4. **Privacy:** We define privacy as the prediction accuracy for the set of sensitive links S by a classifier C_1 which is trained by graph embedding to predict links in a social network graph.

Definition 5. **Utility:** We define utility as the amount of information to be preserved in the graph embedding from an original graph $\mathbf{G}$, which is measured by the non-sensitive link classification accuracy using classifier C_1, and the node classification accuracy using classifier C_2.

Thus, our proposed method is expected to derive a privacy-preserving graph embedding

that can achieve a desired privacy-utility trade-off between privacy and utility.

3.3 Proposed Work

We design the Link-Privacy Preserved Graph Embedding (LPPGE) framework based on Graph Auto-Encoder (GAE) [17] shown as Figure 3.1 which makes use of graph structure $\mathbf{A}$ and node content $\mathbf{X}$ to learn a latent representation $\mathbf{Z}$, and then reconstructs $\hat{\mathbf{A}}$ from $\mathbf{Z}$.

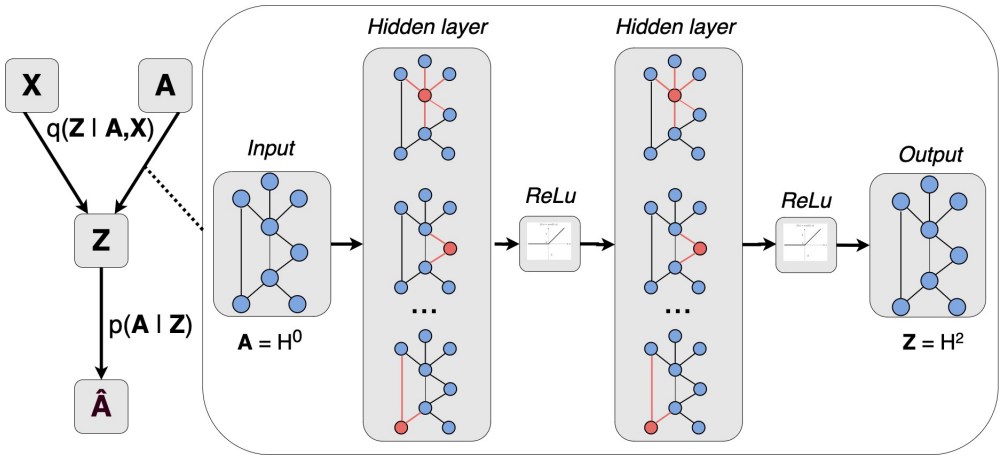


Figure 3.1 Graph Auto-Encoder with Graph Convolutional Networks.

We also utilize a supervised learning mechanism in Adversarial Auto-Encoder (AAE) [91] to achieve privacy protection. Classical AAE forces the latent code to match the previous distribution through the adversarial training module, which distinguishes whether the current latent code $\mathbf{z_i} \in \mathbf{Z}$ comes from the encoder or the previous distribution. While in supervised AAE shown as Figure 3.2, a label vector $\mathbf{z_p}$ is provided to the decoder along with the latent code $\mathbf{z_i}$ to reconstruct the information. The encoder must disentangle some information from $\mathbf{z_i}$ to make $\mathbf{z_i}$ to obey the previous distribution, while the decoder can gain the label information from $\mathbf{Z_p}$, so that the label information can be disentangled from $\mathbf{z_i}$ during the

reconstruction.

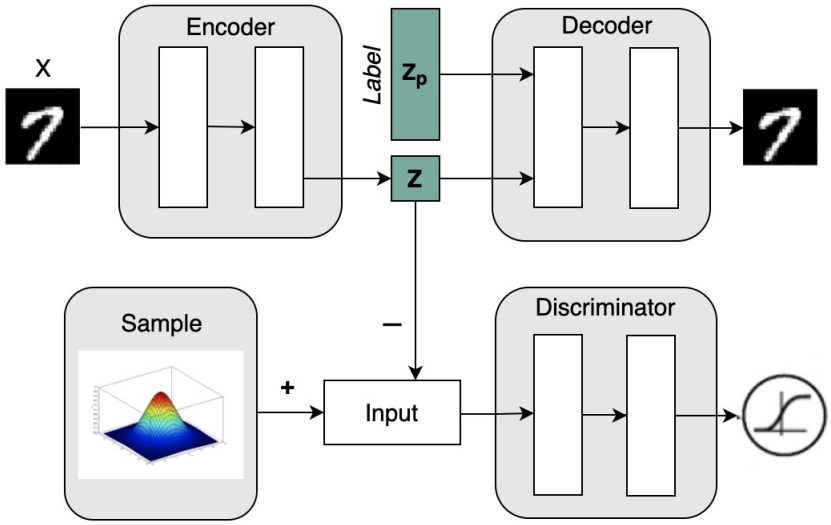


Figure 3.2 Supervised Adversarial Auto-Encoders.

3.3.1 Prepossessing

As shown in Figure 3.3, there are two prepossessing steps before applying the adversarial learning scheme.

First, we delete all the sensitive links in the original graph $\mathbf{G}$ to obtain a modified graph $\mathbf{G}_{train}$ as the input of Algorithm 2 to avoid computing sensitive link prediction loss during reconstruction, which can remove the first-order sensitive link information from the graph embedding.

Second, a privacy embedding is generated via another privacy graph $\mathbf{G}_{priv}$. Because we want to exclude the private information in the graph embedding, a privacy label that represents that information explicitly should be provided to the decoder. However, each embedding vector is a representation of each node, not each link, then how can we add

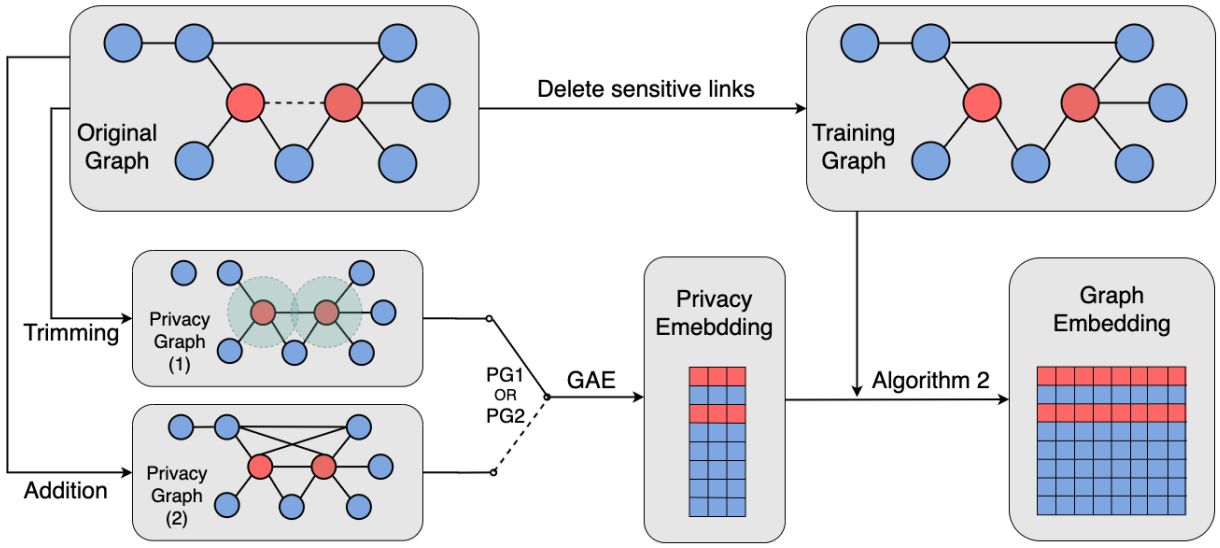


Figure 3.3 Graph Prepossessing. The red nodes are sensitive users ($\lambda = 4$) and the blue ones are non-sensitive users. In the original graph, the dotted line is the sensitive link which is deleted in the training graph. In the privacy graph (1) two links out of radius ($R = 1$) are deleted. In the privacy graph (2) two links are added between sensitive and non-sensitive users. The privacy embedding will be generated from either privacy graph.

a privacy label on each node to include the link information is a challenge. Here, different from the independent one-hot label information in supervised AAE, our private information is located in a pair of nodes' embedding vectors (*i.e.*, a link is determined by two nodes' labels). Note that the private information is not only the direct sensitive link between sensitive users but also includes the high-order information of the sensitive users (*e.g.*, mutual friends and user profiles) which can be used to infer the first-order friendship. Therefore, instead of deleting all the non-sensitive links in $\mathbf{G}$, we develop two methods to generate the privacy graph $\mathbf{G}_{priv}$ separately:

(i) **Trimming method**: we define a radius R for a sensitive user and only keep the links within R in $\mathbf{G}_{priv}$ (*e.g.* when $R = 1$, only the links that connect sensitive users will be kept;

when $R = 2$, only the links that connect sensitive users or their neighbors will be kept).

Algorithm 1 Generate Privacy Graph by Addition Method

Input: $\mathbf{G} = \mathbf{(V, E, X)}$: the original graph
$\mathbf{SE}$: the sensitive links in $\mathbf{G}$
N: the number of $\mathbf{SE}$
M: the number of links to be added
D: the average node degree of sensitive users
$d(i)$: the node degree of user $u_i \in \mathbf{V}$
$\mathbf{se}(i, j)$: the sensitive link between users u_i and u_j
Output: the privacy graph $\mathbf{G}_{priv}$
1: **for** $\mathbf{se}(i, j) \in \mathbf{SE}$ **do**
2: Compute the upper-bound $U = \frac{M}{N} \times \frac{d(i)+d(j)}{2D}$
3: **if** $\exists$ u_i's friend u_x who is 2 hops away from u_j without passing u_i **then**
4: link u_x to u_j **if** the number of new links added for $\mathbf{se}(i, j) \leq U$
5: **end if**
6: **end for**
7: **return** the modified $\mathbf{G}$ as the privacy graph $\mathbf{G}_{priv}$

(ii) **Addition method**: to enhance relationship between sensitive users, we add M links in $\mathbf{G}$ by Algorithm 1. In Step 2 of Algorithm 1, We set an upper-bound U for the number of links to be distributed near each sensitive link $\mathbf{se}(i, j)$ based on the node degrees of its associated sensitive users u_i and u_j. Because we consider higher-degree users as more sensitive, a stronger relationship needs to be built between them. In Step 3 and Step 4 of Algorithm 1, we connect u_j with u_i's friends conditionally, so that more private information can be embedded into the privacy graph embedding.

At last, we compute the privacy graph embedding $\mathbf{Z_p}$ of $\mathbf{G}_{priv}$ generated from either method and use it as a privacy label which describes the relationships of sensitive users precisely.

3.3.2 The Encoder Model

Based on the design of GAE and AAE, LPPGE consists of an encoder, a decoder, and a discriminator which are shown in Figure 3.4. The encoder involves GCNs which extends the operation of convolution to graph data in the spectral domain, and learns a layer-wise transformation by spectral convolution function:

$$\mathbf{Z}^{(l+1)} = f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}), \tag{3.1}$$

where $\mathbf{Z}^{(l)}$ is the input for convolution and $\mathbf{Z}^{(l+1)}$ is the output after convolution. We have $\mathbf{Z}^0 = \mathbf{X} \in \mathbb{R}^{n \times m}$ (n nodes and m features) for our problem. The symmetrically normalized graph Laplacian is applied in $f(\cdot)$ as:

$$f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}) = \phi(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(l)}\mathbf{W}^{(l)}), \tag{3.2}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ ($\mathbf{I}$ is the identity matrix of $\mathbf{A}$) and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ (*i.e.*, the diagonal node degree matrix of $\tilde{\mathbf{A}}$), and ϕ is an activation function.

In LPPGE, we use a 2-layer GCN to extract latent code $\mathbf{Z}'$ from graph $\mathbf{G}_{train}$. The activation function of the first layer is $Relu(\cdot)$ and the second layer is a linear function. In order to disentangle the private information from latent code $\mathbf{Z}'$, we apply a privacy embedding on $\mathbf{Z}'$ before feeding it to the decoder. In this way, the encoder learns a compressed $\mathbf{Z}'$, which excludes the private information but is sufficient for the decoder to reconstruct the graph data because of merging privacy label encoding. Although a lower dimensional $\mathbf{Z}'$ can

extrude more private information, it will also lose some utility information to reconstruct the graph. To moderate the performance decrements, we use the method presented in this paper [92], which is to map the low dimensional representation $\mathbf{Z}'$ to a higher dimensional $\mathbf{Z}$ via a fully connected layer to restore the utility information. Then we can concatenate the privacy embedding with $\mathbf{Z}$ to obtain $\mathbf{Z}^+$ as the input of the decoder.

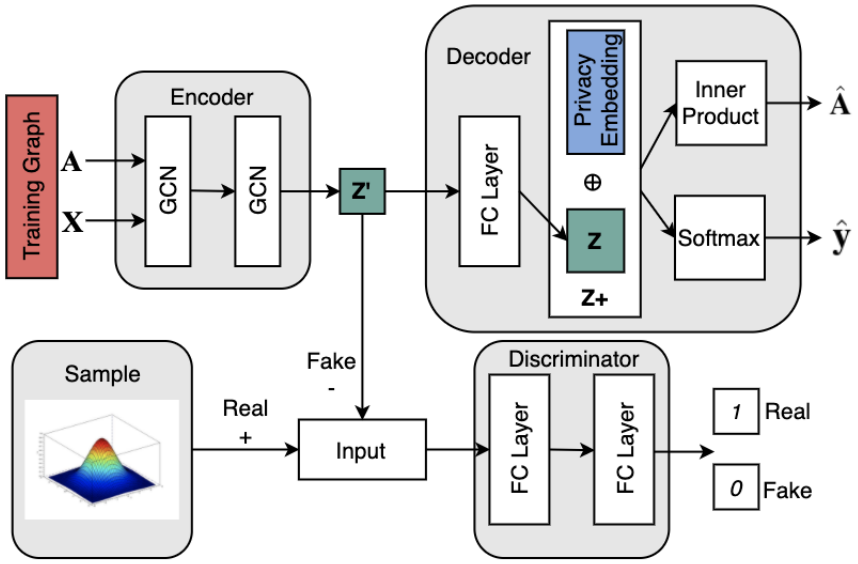


Figure 3.4 The architecture of the adversarial learning in LPPGE.

3.3.3 The Decoder Model

Because the final embedding result should be able to reconstruct $\mathbf{G}_{train}$ with adjacency matrix (structure) $\mathbf{A}$ and content information $\mathbf{X}$, there are two modules in the decoder. The first module reconstructs an adjacency matrix $\hat{\mathbf{A}}$ via the inner product of embedding matrix $\mathbf{Z}^+$ as

$$\hat{\mathbf{A}} = \sigma((\mathbf{Z}^{+})(\mathbf{Z}^{+})^{T}) \tag{3.3}$$

where $\sigma(\cdot)$ is the logistic sigmoid function.

The cross-entropy loss between $\hat{\mathbf{A}}$ and $\mathbf{A}$ would be minimized as the loss for link prediction:

$$L_{link} = -\frac{1}{N^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{A}_{ij}log(\hat{\mathbf{A}}_{ij}), \tag{3.4}$$

where $\mathbf{A}_{ij}$ and $\hat{\mathbf{A}}_{ij}$ are the corresponding elements of $\mathbf{A}$ and $\hat{\mathbf{A}}$.

The second module is a category classifier which decodes $\mathbf{Z}^{+}$ using a soft-max function $\hat{\mathbf{y}}_i = softmax(\mathbf{z}_i^{+})$ and computes the cross-entropy loss between the one-hot label $\mathbf{y}_i$ of each user. The loss function is defined as:

$$L_{label} = -\frac{1}{N}\sum_{i=1}^{n}\mathbf{y}_i log(\hat{\mathbf{y}}_i) \tag{3.5}$$

Thus the total loss of LPPGE is the combination of the link prediction loss and the category classification loss:

$$L_{recon} = L_{link} + \alpha L_{label}, \tag{3.6}$$

where α is a trade-off parameter between the link prediction loss and the category classification loss.

3.3.4 The Discriminator Model

The discriminator consists of two fully connected layers. It will be trained to distinguish whether a latent code is from the Gaussian distribution (positive) or from the encoder of LPPGE (negative). We optimize the discriminator by minimizing the following loss:

$$L_{dc} = -log(D(\mathbf{x})) - log(1 - D(\mathbf{z}'_i)), \tag{3.7}$$

where $D(\mathbf{x})$ is the discriminator's estimation of the probability that real sample $\mathbf{x}$ is from the Gaussian distribution, and $D(\mathbf{z}'_i)$ is the discriminator's estimation of the probability that a latent code $\mathbf{z}'_i$ is real. During the optimization, the graph embedding $\mathbf{Z}$ is regularized to Gaussian distribution.

Algorithm 2 Link-privacy Preserved Graph Embedding

Input: $\mathbf{G = (V, E, X)}$: the original graph
T: the number of training iterations
d: the dimension of the final graph embedding $\mathbf{Z}$
Output: $\mathbf{Z} \in \mathbb{R}^{n \times d}$

Generate the training graph G_{train} and the privacy graph G_{priv} from G
Generate the privacy embedding $\mathbf{Z}_p$ from G_{train}
for $iteration = 0$ to T **do**
    Generate the latent code $\mathbf{Z}'$ of G_{train} by **Eq. 3.2**
    Map $\mathbf{Z}'$ to the higher dimensional $\mathbf{Z}$
    Concatenate $\mathbf{Z}_p$ with $\mathbf{Z}$ as the input of the decoder
    Update the encoder and decoder by minimizing **Eq. 3.6**
    Update the discriminator by minimizing **Eq. 3.7**
    Update the encoder by maximizing **Eq. 3.7**
end for
return $\mathbf{Z} \in \mathbb{R}^{n \times d}$

3.3.5 Algorithm Explanation

We summarize the proposed framework in Algorithm 2. The framework has two stages: prepossessing (Step 1 and Step 2) and training (Step 3 to Step 10). In Step 8, we update the discriminator to tell if the input is from a positive sample or the graph encoder. In Step 9, the encoder is updated to confuse the discriminator. These two steps can be integrated as $\min\max L_{dc}$ in Eq. 3.7. After T epochs' training, we return the final link-privacy preserved graph embedding $\mathbf{Z} \in \mathbb{R}^{n \times d}$ in Step 11.

3.4 Experiments

In our experiments, we employ four different datasets summarized in Table 3.1. The first two datasets are **Cora** and **Citeseer** used in the GAE paper [17], and both of them consist of scientific publications as nodes and citation relationships as edges. The features are unique words in each document. The other two datasets are two ground truth social network datasets: **Yale** and **Rochester**, which are composed of Facebook users from Yale University and Rochester University.

By default, we assume the number of sensitive links is around 1% of the total links, then we find the corresponding λ to define the sensitive users and sensitive links in each dataset. For the addition method, we set the target number M to be 10% of the total links and the actual number is around 9%.

We compare LPPGE with the following baselines:

1. **GAE** is an autoencoder-based model for unsupervised learning on graph-structured

Parameters	Cora	Citeseer	Yale	Rochester
Nodes	2,708	3,327	8,758	4,563
Links	5,429	4,732	405,450	167,653
Features	1,433	3,703	7	7
λ	13	16	320	220
Sen_Links	53	62	2,800	1,446
Per*	0.97%	1.31%	0.69%	0.86%
Radius R	2	1	2	2
α	3	3	2	2
M	540	470	40,000	16,000
Added_Links	405	434	39,200	15,906

Table 3.1 The employed graph datasets and the parameters of each dataset. $Per*$ is the percentage of the sensitive links in the graph. The actual number of added links (Addition method) is less than M because of the graph structure and upper-bound U round-down for each sensitive link.

data.

2. **GAE_RM** uses the same framework as GAE but deletes the defined sensitive links.

3. **DPNE** is a differentially private network embedding method based on DeepWalk as matrix factorization. It applies the objective perturbation approach by adding noise in the objective function of matrix factorization to learn a representation satisfying differential privacy.

Approaches	Cora		Citeseer	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	81.8 ± 1.0	85.6 ± 3.5	83.7 ± 1.4	92.1 ± 3.4
GAE_RM	84.6 ± 0.7	83.3 ± 3.0	87.5 ± 1.2	91.8 ± 3.3
DPNE	55.3 ± 1.3	67.3 ± 4.2	52.7 ± 1.3	67.5 ± 5.2
NPGE	85.1 ± 1.4	89.2 ± 2.8	88.9 ± 1.5	92.6 ± 2.6
LPPGE(T)	81.5 ± 1.1	75.8 ± 3.9	85.6 ± 1.6	81.0 ± 3.9
LPPGE(A)	**80.5 ± 1.2**	**72.3 ± 3.1**	**84.1 ± 1.2**	**77.0 ± 3.5**
Approaches	Yale		Rochester	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	84.4 ± 1.4	90.0 ± 1.2	85.1 ± 0.4	90.7 ± 1.1
GAE_RM	84.2 ± 0.2	89.6 ± 0.8	84.7 ± 0.3	88.5 ± 0.7
DPNE	48.8 ± 10	52.5 ± 10	60.6 ± 1.2	75.9 ± 3.9
NPGE	81.7 ± 0.1	85.2 ± 0.6	83.5 ± 0.4	88.0 ± 1.2
LPPGE(T)	81.2 ± 0.2	83.1 ± 0.5	82.2 ± 0.3	83.1 ± 0.7
LPPGE(A)	**80.2 ± 0.2**	**80.7 ± 1.0**	**80.4 ± 0.3**	**81.3 ± 1.2**

Table 3.2 Results of link prediction.

Approaches	Cora(7)		Citeseer(6)		Yale(6)		Rochester(2)	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	74.0	71.5	58.6	54.0	85.8	87.0	84.8	84.1
GAE_RM	75.1	71.3	63.5	55.5	85.1	86.3	85.8	84.4
DPNE	14.5	6.63	18.6	8.32	24.2	4.66	50.2	44.7
NPGE	72.4	68.0	69.0	64.1	78.9	77.6	84.1	80.0
LPPGE(T)	79.2	70.8	56.4	47.7	84.8	83.5	86.2	82.8
LPPGE(A)	73.9	71.8	66.9	62.3	84.0	83.3	86.4	83.6

Table 3.3 Results of node classification. $Dataset(*)$ indicates the number of the categories in each dataset.

4. **Non-privacy Graph Embedding (NPGE)** uses the same architecture as LPPGE, but the privacy embedding of NPGE is generated from the graph that only has sensitive links.

For LPPGE, we implement **LPPGE(T)** using the trimming method to generate the privacy graph and **LPPGE(A)** using the addition method. We set the embedding size of DPNE as 64 dimensions with a relatively large privacy budget $\epsilon = 1$ to maximize the defined utility for all the datasets. For the rest of the methods, we embed a graph into a 16-dimensional space for the Cora and Citeseer datasets, while a 32-dimensional space for the Yale and Rochester datasets. In LPPGE, we also set hidden code $\mathbf{z}'$ to be a 4-dimensional vector for all the datasets. The performance of LPPGE is evaluated on two aspects which are link prediction and node classification.

3.4.1 Link Prediction

We train a multi-layer perceptron (MLP) classifier as the attacker, which tries to predict the sensitive links in a social network by graph embedding. The input of the MLP is the embedding vectors of two users in the social network and the output is the relationship

between these two users. We assume 10% of non-sensitive links are exposed to the attacker as the positive samples in the training set, and the same number of negative samples are collected by randomly selecting unconnected users. We present the results in the Macro F1-score of the non-sensitive links and the sensitive links separately. We conduct each experiment 10 times and show the mean values with standard errors as the final scores. The details of the experiment results on link prediction are shown in Table 3.2.

The performance of GAE_RM shows that even if the sensitive links are deleted, the attacker is still able to predict its existence precisely according to the high-order information. Although DPNE has the lowest prediction accuracy for sensitive links, its prediction accuracy of non-sensitive links is much lower than other methods. It can be seen DPNE costs a significant amount of utility information in its embedding to preserve privacy, and differential privacy is susceptible to inference attacks. The result of NPGE shows that merely extruding sensitive links in graph embedding is not enough to protect sensitive information. Moreover, it should be noted that as sensitive users have large degrees, in the traditional graph embedding methods, sensitive links are easier to predict than non-sensitive links, which can be observed in Table 3.2. It is also worth mentioning that LPPGE(T) and LPPGE(A) both can reduce sensitive accuracy much lower than non-sensitive ones on the Cora and Citeseer datasets. For the Yale and Rochester datasets, LPPGE(A) can reduce sensitive accuracy by about 10% with only losing 5% utility accuracy on non-sensitive links. These facts prove that LPPGE is capable of reducing the attacker's prediction accuracy of sensitive links while slightly sacrificing the utility of embedding to reconstruct the non-sensitive part of a graph.

3.4.2 Node Classification

We apply two classifiers, MLP and Support Vector Machine (SVM), to predict user category labels through graph embedding. For all the datasets, 5-fold cross-validation is used to ensure the model's reliability and effectiveness, and the results are given as Macro F1-score in Table 3.3.

Because DPNE only embeds graph structure information without node attributes in the embedding, even though with a higher dimensional embedding space, the performance is still poor on the classification task and the graph embedding utility is low. Comparing LPPGE with GAE and GAE_RM, the classifiers have similar accuracy in predicting node's class labels on all the datasets, which indicates LPPGE can maintain accurate cluster information at the same level as privacy protection.

3.4.3 Trade-off Between Utility and Privacy

To demonstrate the trade-off between utility and privacy, we compute the ratio of utility (*i.e.*, the sum of the prediction accuracy of non-sensitive links and node classification) to privacy (*i.e.*, the prediction accuracy of sensitive links). As shown in Figure 3.5, both LPPGE(T) and LPPGE(A) can achieve better performance on the aspects of privacy protection and data usability preservation. Due to space limitations, the experiment results for link prediction and node classification are presented in the Appendix.

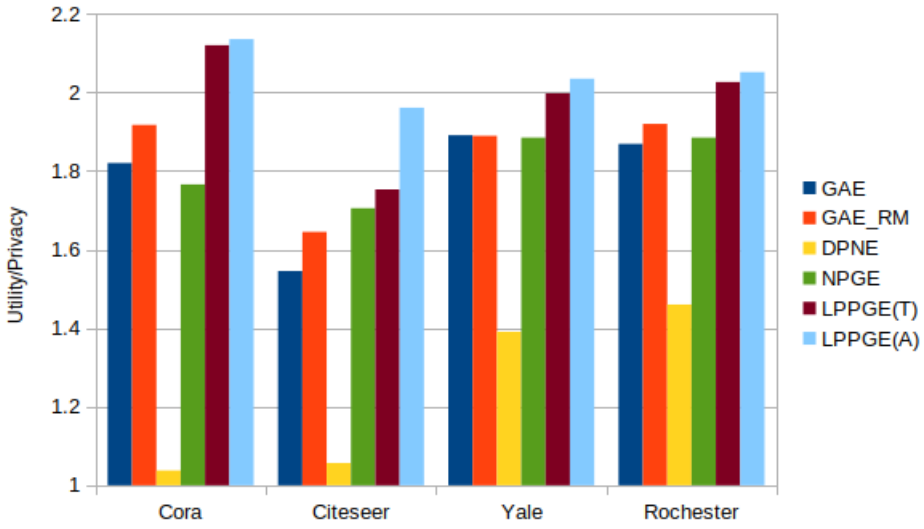


Figure 3.5 Evaluation of trade-off between utility and privacy.

3.4.4 Customization

In real-world situations, data owners may only need to protect some specified relationships between sensitive users based on their demands. Hence, we randomly select 25%, 50%, and 75% links from pre-defined sensitive links as the new sensitive links and conduct the same Utility/Privacy evaluation to testify to the scalability of our model. The results for different scales shown in Figure 3.6 demonstrate that LPPGE can fulfill customized requests for privacy protection.

3.4.5 Graph Visualization

We visualize the Cora and Yale datasets in 2-dimensional space by applying the t-SNE algorithm to the learned embedding. The Cora dataset is partitioned by the publication subject and the Yale dataset is partitioned by the user's class year. Each subgroup is represented in a different color. The results shown in Figure 3.7 and Figure 3.8 validate our

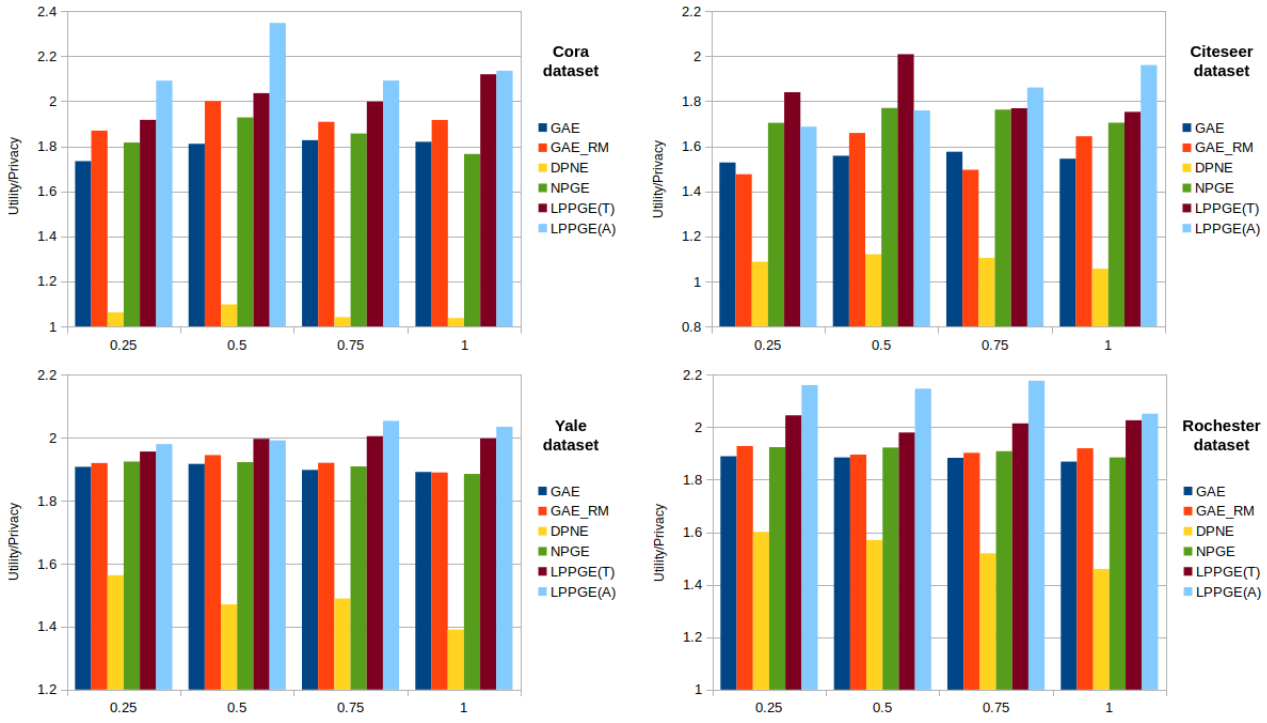


Figure 3.6 Evaluation of trade-off between utility and privacy for different scales. The X-axis is the percentage of sensitive links.

assertions in the node classification section through a meaningful layout.

3.5 Conclusions

In this study, we investigate a pertinent privacy issue associated with social graph embedding and introduce an innovative graph embedding framework. Through the combined use of a graph autoencoder and adversarial learning, our method regularizes the latent representation to align with a predetermined distribution, effectively eliminating sensitive information from the graph embedding. Experimental outcomes show that our approach successfully balances privacy protection with data utility, outperforming alternatives that do not prioritize pri-

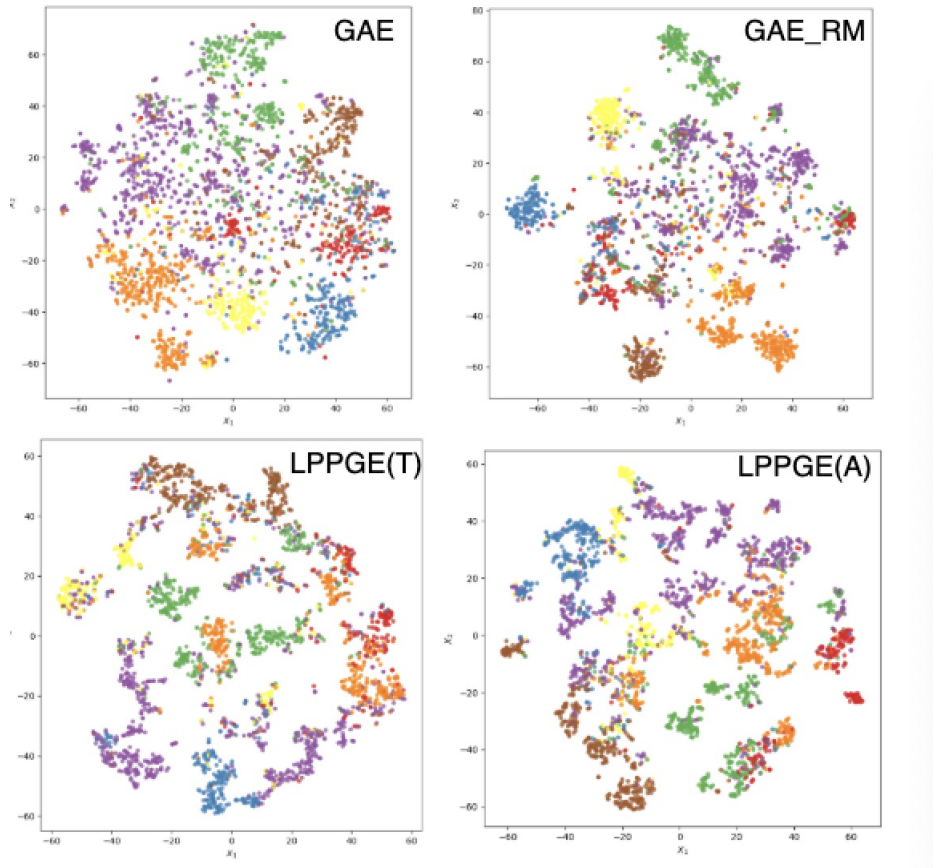
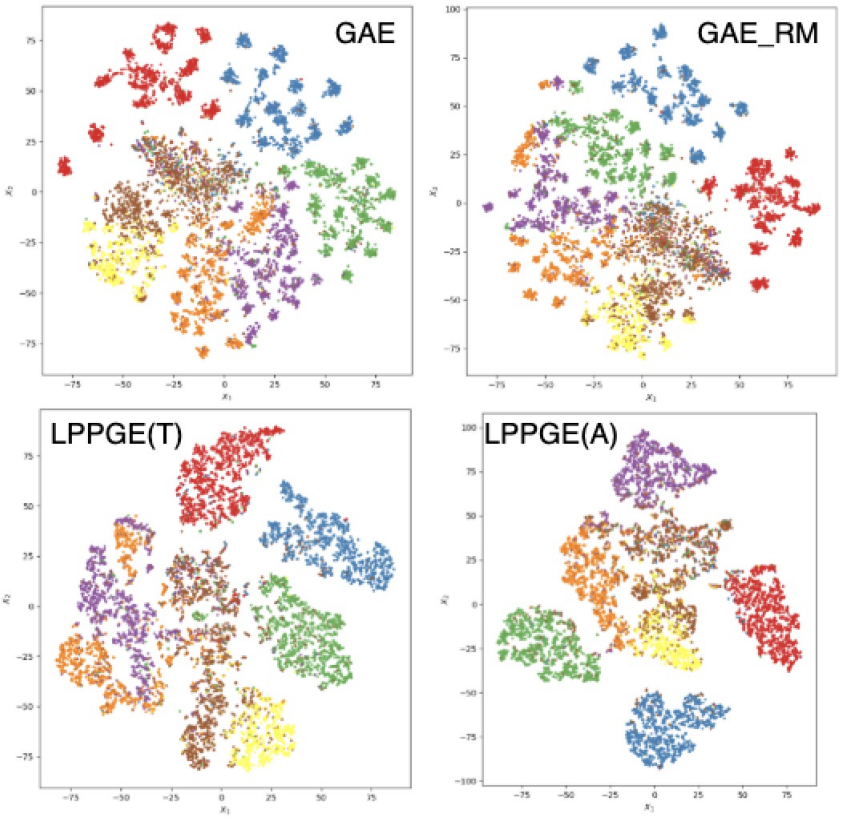


Figure 3.7 Visualization comparison for the Cora dataset. Methods from left to right, top to bottom are GAE, GAE_RM, LPPGE(T), and LPPGE(A).

vacy. Looking ahead, our interest lies in exploring dynamic graph embedding's potential applications, a burgeoning research avenue. The dynamic nature of graph sequences allows adversaries to potentially deduce or reconstruct sensitive details, posing significant challenges in the realm of data privacy and security.



C

Figure 3.8 Visualization comparison for the Yale dataset. Methods from left to right, top to bottom are GAE, GAE_RM, LPPGE(T), and LPPGE(A).

CHAPTER 4

Privacy-Preserving Federated Graph Neural Network Learning on Non-IID Graph Data

4.1 Introduction

Data providers often share data to bolster the analytical performance across participants. However, this collaboration can inadvertently compromise the privacy of data owners. Simply put, insecure data sharing combined with inadequate de-anonymization is akin to freely handing out the owner's information [93]. Federated Learning (FL) [94] offers an alternative learning approach that sidesteps centralized data collection. Traditional server models might inadvertently expose sensitive user information, an issue FL circumvents by training deep neural networks across localized datasets without sharing data samples with the central server or other clients [95, 96].

Graph data proves invaluable in tasks that deal with intricate relationships and dynamic schematics, such as in block chain management [97] or recommendation systems. While graph neural networks have made strides using representation learning for tasks like node classification and link prediction [98], multiple barriers inhibit FL's widespread application within the realm of graph neural networks. Notably, the non-IID nature of graphs [99] suggests that, in the context of expansive and noisy real-world graphs, FL might underperform compared to centralized methods. There is a potential for GNNs to overfit extensive training datasets unless they undergo proper regularization [100]. Additionally, FL's aggregation mechanism might stumble on sparse graphs where local neighborhood nodes contribute more

noise than useful data for feature aggregation [101]. The wide-ranging diversity in GNN models results in a lack of uniformity in current federated GNN definitions [102]. Many existing FL algorithms, including the FedAvg [103], primarily cater to IID datasets, making the fusion of information across diverse clients in federated GNNs challenging [104]. Especially in scenarios where clients possess distinct sample nodes and can't share comprehensive topology data due to privacy concerns, deploying traditional averaging strategies in the federated framework becomes ill-suited, given the non-uniform input nodes in graph neural networks.

To address these challenges, we introduce a unique federated learning framework tailored for graph neural networks, incorporating an embedding alignment technique. As the framework only amalgamates client-approved public data, it substantially curtails privacy risks during the learning process. This alignment technique ensures that clients with non-IID data can effectively exchange information. Moreover, our empirical findings indicate that infusing aligned information into local models acts as a form of regularization, mitigating the potential for overfitting. The main contributions of our work are summarized as follows:

- We investigate a general training scenario of the federated GNNs setting in which multiple clients hold non-IID graph datasets sharing partial structural equivalence.
- We propose a novel framework to integrate federated learning and embedding alignment techniques into an end-to-end process flow to obtain accurate embedding results for individual clients.
- We conduct extensive experiments on ground truth datasets to prove the effectiveness of the proposed method with the embedding alignment technique and demonstrate the

competitive performance of PPFL-GNN framework with respective to noise resistance.

4.2 Proposed Work

In this section, we first introduce the problem formulation of our work, and then explain the details of our approach to learning graph representation in a privacy-preserving way based on two state-of-the-art models.

4.2.1 Problem Statement

Denote $\mathbf{C} = \{c_1, c_2, ...c_n\}$ as the sets of clients participating in federated learning, and client c_i holds a local undirected graph $\mathcal{G} = (\mathbf{U, E, F})$ including node set $\mathbf{U}$, edge set $\mathbf{E}$, and node-feature set $\mathbf{F}$. We assume all the local graphs share a certain amount of nodes defined as a public node set $\mathbf{U}_k = \mathbf{U}_1 \cap \mathbf{U}_2 \cap ... \cap \mathbf{U}_n$. To protect privacy, each client saves the original data locally, including the edge and attribute information of non-public nodes. Only the processed public node information, which is generated as public node embedding by the client's local model, will be uploaded to the server. Our goal is to generate accurate node representation for each client by utilizing federated learning without building and storing the entire graph on the server or client.

4.2.2 Federated DeepWalk

DeepWalk extends the idea of language modeling to network topology [12], which forms the embryo of graph embedding. Given a random walk sequence composed of network nodes:

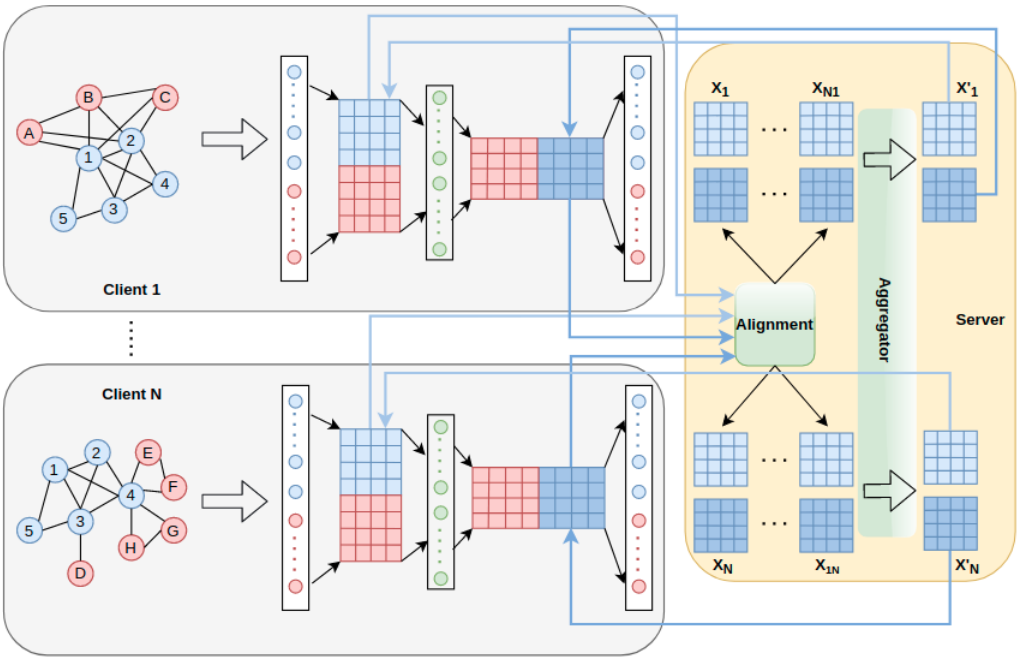


Figure 4.1 Overview of the Federated DeepWalk framework. The red nodes are private, and the blue nodes are public. Local training is highlighted in grey, and server aggregation is highlighted in yellow.

$$V_1^n = (v_0, v_1, ..., v_n) \tag{4.1}$$

where $v_i \in \mathbf{U}$. The goal so far is to retrieve the likelihood of observing v_i given the previous $i-1$ nodes in the random walk:

$$\Pr(v_i|(v_1, v_2, ..., v_{i-1})) \tag{4.2}$$

To learn the latent representation, instead of only a probability distribution of node co-occurrence, DeepWalk introduces a mapping function $\Phi : v \in V \mapsto \mathbb{R}^{|V| \times d}$, which actually is a $|V| \times d$ matrix of free weights serving as the low-dimensional representations of all network

nodes in the graph.

However, the computation is not efficient depending on the length of the random walks. Thus, the SkipGram method in Word2vec [13] is applied to solve the computational problem. Rather than predicting the occurrence of a missing node in the walk, we compute the likelihood of a node appearing as a neighbor in a given window, and the new optimization goal is summarized as follows:

$$\min_{\Phi} -\log \Pr(\{(v_{i-w}, ..., v_{i-1}, v_{i+1}, ..., v_{i+w})\} | \Phi(v_i)) \tag{4.3}$$

where w is the window size for iterating the possible collocation of the given node v_i. Suppose we deploy DeepWalk as the neural network model in the federated learning setting, then the local client c_i can train a low-dimensional latent representation $\mathbb{R}_i^{|V| \times d}$ of his local graph $\mathcal{G}_i$. After all clients have generated their local graph embeddings, the challenge of a federated learning setting is how all clients collaborate to improve the training results with less disclosure of sensitive information.

In traditional federated learning, each client uploads all weights of the local model to a central server. The central server aggregates these weights to update the global model and then distributes the global model back to the clients. However, in our problem definition, we cannot aggregate all weights directly because each client holds a different subgraph of the global network, which means the trained latent representations only share commonality on the public nodes partially. Because the potential relationship between public and private nodes are stored in the public nodes' latent representations, as shown in Figure 4.1, instead

of uploading all weights (*i.e.*, the latent representations of all the nodes in the local graph), a client can only upload the weights related to the public nodes (*i.e.*, the latent representations of public nodes), which also carry some sensitive information of the private nodes.

Since the latent representations of public nodes are generated from different training graphs, simple aggregation and distribution will break their connections with the unprocessed latent representations of private nodes on the local client. Thus, we apply an embedding alignment technique in the weight aggregation on the central server to convert the latent representations from other clients into a form that the local client understands. For example, there are two local clients c_x and c_y sharing k nodes in the graph. Let $X = \{\Phi_x(u_1), ..., \Phi_x(u_k)\}$ and $Y = \{\Phi_y(u_1), ..., \Phi_y(u_k)\}, u_k \in \mathbf{U}_k$ be two sets of k public node embeddings coming from c_x and c_y respectively. For c_y to understand the information of X, we need to align/translate X into the space of c_y, which technically is using a linear mapping matrix W that maps X from the source space c_x to the target space c_y. Furthermore, we can encapsulate the problem to the Procrustes problem [105] and solve it via the Singular Value Decomposition (SVD) of YX^T:

$$\begin{aligned} W^* = \text{argmin}_{W \in M_d(\mathbb{R})} ||WX - Y||_F = UV^T, \\ \text{with } U \sum V^T = \text{SVD}(YX^T) \end{aligned} \tag{4.4}$$

where $M_d(\mathbb{R})$ is the $d \times d$ matrix space of real numbers. We denote $X_y = WX$ as the aligned embeddings from source space c_x to target space c_y and Y_x in the opposite way. The server aggregates X_y and Y to obtain a merged weight Y' and returns Y' to c_y for substituting

the current public node embedding vector $\Phi(y_k)$. For multiple clients $\mathbf{C} = \{c_1, c_2, ...c_n\}$, the server aligns the embeddings from any pair of clients $\forall c_i, c_j \in \mathbf{C}$ and applies the average aggregation on all the aligned embeddings in the same client's space to get the returning updates for each client. The local clients use the updates as the initial weights to train in a new round. Algorithm 3 summarizes the complete training procedure.

4.2.3 Federated GAT Framework

GAT [106] introduces an attention mechanism to replace the statically normalized convolution operation in GCN [107]. The input to a single attentional layer is a set of node features, $\mathbf{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_n}\}, \overrightarrow{h_i} \in F$, where n is the number of nodes, and F is the node feature set. A linear transformation is firstly applied to every node feature for higher-level expression:

$$z_i^{(l)} = W^{(l)} h_i^{(l)} \tag{4.5}$$

where $W^{(l)}$ is a learn-able weight matrix.

Different from the dot-product attention mechanism in GCN, GAT applies the additive attention mechanism, which concatenates the z embeddings of two neighbors i and j to compute a pair-wise unnormalized attention score $e_{ij}^{(l)}$ between them. The additive attention mechanism takes the dot product of the concatenation and a weight vector $\overrightarrow{a}$, then applies a LeakyReLU activation function. In order to compare the attention scores with different nodes, a normalized coefficient $\alpha_{ij}^{(l)}$ is computed by the softmax function in the end:

Algorithm 3 The Federated DeepWalk Framework

Input: $\mathbf{C} = \{c_1, c_2, ...c_n\}$: the set of clients
$\mathcal{G}_i$: the local subgraph hold by c_i
$\mathbf{U}_k$: the public nodes shared among $\mathbf{C}$
Output: the matrix of node representation $\Phi_i \in \mathbb{R}^{|V| \times d}$ of $\mathcal{G}_i$

Local Clients:
for each client $c_i \in \mathbf{C}$ **do**
  Compute the DeepWalk model weights Φ_i
  Generate the public nodes' embeddings X_i of U_k from Φ_i:
    $X_i = \{\Phi_i(u_1), ..., \Phi_i(u_k)\}$
  Upload X_i to the server
end for

while not converge **do**
  Server:
  for each $i \in k$ **do**
    for each $j \in k (i \neq j)$ **do**
      Align X_j into c_i's space: $X_{ji} = W_{ji} X_j$
    end for
    Aggregate all the aligned embeddings with X_i
      $X_i' = \frac{1}{k}(\sum_j^k X_{ji} + X_i)$
    distribute X_i' to client c_i for local update
  end for

  Local Clients:
  for each client $c_i \in \mathbf{C}$ **do**
    Substitute the public nodes' embeddings in Φ_i by X_i'
      $\Phi_i' \leftarrow (\Phi_i, X_i')$
    Initial the DeepWalk model with Φ_i'
    Compute the model weights Φ_i
  end for
end while
return the matrix of node representation $\Phi_i \in \mathbb{R}^{|V| \times d}$ of $\mathcal{G}_i$

$$\begin{aligned} \alpha_{ij}^{(l)} &= \text{softmax}_j(e_{ij}^{(l)}) \\ &= \frac{\exp\big(\text{LeakyReLU}(\overrightarrow{a}^T [z_i^{(l)} || z_j^{(l)}])\big)}{\sum_{k \in \mathcal{N}_i} \exp\big(\text{LeakyReLU}(\overrightarrow{a}^T [z_i^{(l)} || z_k^{(l)}])\big)} \end{aligned} \tag{4.6}$$

where $\mathcal{N}_i$ is some neighbor of node i in the graph, $||$ denotes the concatenation operation, and

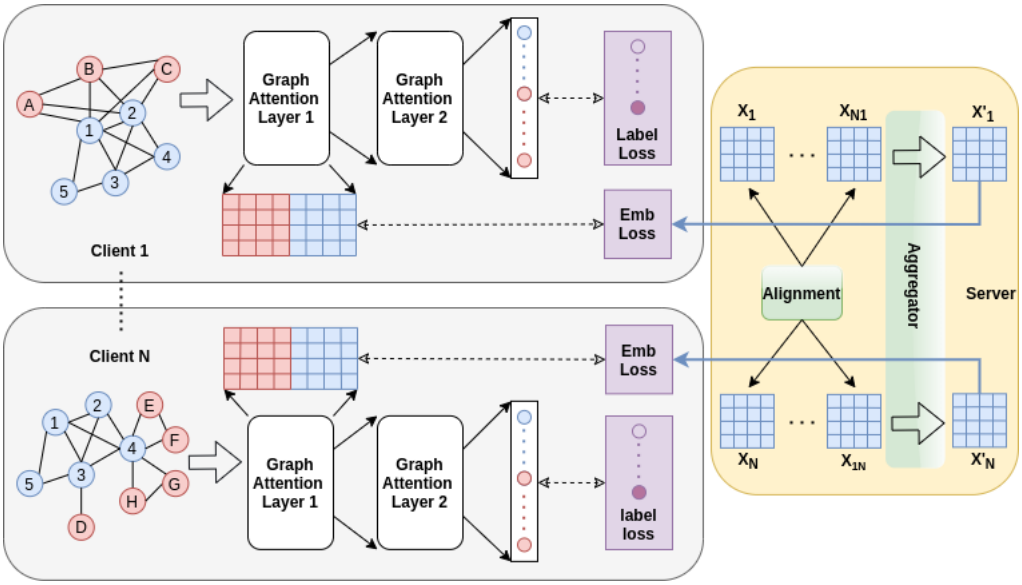


Figure 4.2 Overview of the Federated GAT Framework. The red nodes are private, and the blue nodes are public. Local training is highlighted in grey, and server aggregation is highlighted in yellow.

$\cdot^T$ represents transposition. Having the normalized attention coefficients calculated, GAT generates the next-level embedding of node i by aggregating its neighbors' embeddings, scaled by the attention coefficients.

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l)} z_j^{(l)}\right) \tag{4.7}$$

Once we obtain the local embeddings, we have to face the similar challenge of collaborating with different clients in federated learning as the Federated DeepWalk framework. Although in DeepWalk model, we can extract the public nodes' embeddings from the model weights directly, the weights of GAT integrate both public and private information and cannot be split directly by node's category. Therefore, as shown in Figure 4.2, we upload the public nodes' embeddings X coming from the model's intermediate layer ($i.e.$, $h_i^{(l)}, i \in \mathbf{U}_k$)

to the server without exposing the model weights. Then, the server executes the same processes in the Federated DeepWalk framework to align, aggregate, and distribute the updates X' to clients. As we cannot use the aligned embedding to manipulate GAT's model weights directly, another cosine-embedding loss $\mathcal{L}_{emb}$ is added beside the original cross-entropy loss $\mathcal{L}_{label}$ to integrate the information of X' back into the model.

$$\mathcal{L}_{emb} = 1 - cos(X, X') \tag{4.8}$$

$$\mathcal{L}_{label} = -\frac{1}{N}\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{4.9}$$

where y_i is the one-hot label of each node and $\hat{y}_i = softmax(h_i^{(l+1)})$. Thus, the new loss of local training is the combination of the cosine-embedding loss and the cross-entropy loss:

$$\mathcal{L}_{new} = \mathcal{L}_{label} + \beta\mathcal{L}_{emb}, \tag{4.10}$$

where β is a model hyper-parameter to balance the local information preservation and the external information integration. During the experiment, we observe that the last attention layer is so powerful that it overwhelms the cosine-embedding loss of the final output $h_i^{(l+1)}$. Hence, based on Two-stage CNN training introduced [108] and Federated Split learning [109], we inject the external information via the intermediate layer $h_i^{(l)}$ at an earlier stage. Algorithm 4 summarizes the complete procedure of the Federated GAT Framework.

Algorithm 4 The Federated GAT Framework

Input: $\mathbf{C} = \{c_1, c_2, ...c_n\}$: the set of clients
$\mathcal{G}_i$: the local subgraph hold by c_i
$\mathbf{U}_k$: the public nodes shared among $\mathbf{C}$
Output: the node embeddings H_i' of $\mathcal{G}_i$

Local Clients:
for each client $c_i \in \mathbf{C}$ **do**
  Compute the GAT model embedding H_i'
  Generate the public nodes' embeddings X_i of U_k from the intermediate H_i:
    $X_i = \{H_i(u_1), ..., H_i(u_k)\}$
  Upload X_i to the server
end for

while not converge **do**
  Server:
  for each $i \in k$ **do**
    for each $j \in k (i \neq j)$ **do**
      Align X_j into c_i's space: $X_{ji} = W_{ji} X_j$
    end for
    Aggregate all the aligned embeddings with X_i
      $X_i' = \frac{1}{k}(\sum_j^k X_{ji} + X_i)$
    distribute X_i' to client c_i for local update
  end for

  Local Clients:
  for each client $c_i \in \mathbf{C}$ **do**
    Take X_i' as new input weights
    Compute the GAT-model embedding H_i' with loss $\mathcal{L}_{new}$
  end for
end while
return the node embeddings H_i' of $\mathcal{G}_i$

4.3 Experiments

In this section, we present the experiments developed by Pytorch and conducted on a workstation with an Intel Core i7 2.80GHz CPU and an NVIDIA GeForce GTX 1070 GPU.

4.3.1 Datasets

In our experiments, we employ two datasets, Cora [110] and Citeseer [111], which are commonly used in GNNs research. The Cora dataset includes 2,708 publications as nodes classified into seven classes, and its citation network consists of 5429 links. The Citeseer dataset includes 3,327 nodes classified into six classes, and its citation network consists of 4,732 links. Both datasets use unique words in each document as the node features. We set up four clients participating in the federated learning, so each dataset is split into four subgraphs with an equal number of nodes and assigned to each client. By default, each Cora subgraph has 1,489 (55%) nodes in total with 1,083 (40%) public nodes and 406 (15%) private nodes, while each Citeseer subgraph has 1,829 (55%) nodes in total with 1,330 (40%) public nodes and 499 (15%) private nodes. (%) shows the percentage of the nodes in the original graph.

4.3.2 Baselines and Metrics

We use the client's local training as the baseline to verify whether our framework can improve each client's graph embedding result through collaborative training. In the DeepWalk-based training, all clients use the same model architecture with randomly initialized weights for local training or federated training. While in the GAT-based training, clients use the original GAT model for the local training and the modified GAT model with embedding loss for the federated training. Other architecture and parameters are fixed in a controlled experiment.

For all the implementations, we embed each graph into a 16-dimensional space and run the experiments on the classification tasks to evaluate the quality of the embedding by

applying one Multi-layer Perceptron (MLP) classifier and another Support Vector classifier (SVC) implemented in the Python module Scikit-learn to predict the label of a node. For all the experiments, we use 5-fold cross-validation to ensure models' reliability and effectiveness, and the classification results of the two frameworks are given as Micro F1-scores.

Dataset (Classifier)	Client1		Client2		Client3		Client4		GLOBAL
	LOC	FED	LOC	FED	LOC	FED	LOC	FED	
Cora(MLP)	79.1	**80.1**	76.0	76.0	74.3	**76.1**	75.0	**76.6**	+4.41
Cora(SVC)	79.3	**81.4**	77.5	**78.7**	75.5	**77.9**	75.0	**78.0**	+8.72
Citeseer(MLP)	48.1	**51.4**	46.4	**50.4**	48.9	**51.6**	49.2	**51.7**	+12.5
Citeseer(SVC)	56.3	**60.6**	53.5	**58.6**	57.3	**61.6**	55,3	**60.4**	+18.8
Cora(Full)	MLP		79.9		SVC		82.0		
Citeseer(Full)	MLP		58.6		SVC		65.4		

Table 4.1 The Results of the Federated DeepWalk Framework. LOC indicates the result of local training, FED indicates the result of federated learning, GLOBAL indicates the cumulative improvement of all clients.

Dataset (Classifier)	Client1		Client2		Client3		Client4		GLOBAL
	LOC	FED	LOC	FED	LOC	FED	LOC	FED	
Cora(MLP)	84.4	**86.3**	85.7	**85.8**	83.1	**83.8**	85.7	**86.0**	+3.01
Cora(SVC)	86.1	**86.7**	**86.5**	86.0	**85.6**	85.0	85.4	**86.0**	+0.1
Cora_Noise(MLP)	**81.4**	81.0	79.0	**79.5**	72.2	**75.9**	70.6	**77.3**	+10.4
Cora_Noise(SVC)	82.0	**82.4**	80.0	**80.8**	74.5	**77.4**	72.3	**78.3**	+10.1
Citeseer(MLP)	72.3	**74.7**	72.8	**74.1**	72.8	**74.3**	72.7	**75.0**	+7.51
Citeseer(SVC)	72.8	**74.5**	72.3	**74.3**	72.1	**74.3**	72.1	**74.1**	+7.82
Cora(Full)	MLP		86.2		SVC		86.1		
Citeseer(Full)	MLP		74.7		SVC		74.8		

Table 4.2 The Results of the Federated GAT Framework. Cora_Noise is the Cora dataset with noisy labels.

4.3.3 Performance Evaluation

The Federated DeepWalk framework results are presented in Table 4.1. We can observe that in contrast to the embeddings generated by limited local information, both classifiers can achieve higher classification accuracy on the embeddings trained by full use of the graph

data. Under this precondition, the proposed FL methodology can improve the global results by 4.41%(MLP) and 8.72%(SVC) on the Cora dataset, while 12.5%(MLP) and 18.8%(SVC) on the Citeseer dataset, respectively. Specifically, every client in the Citeseer experiment receives steady improvement compared with the local baseline.

For the Federated GAT framework, we set the hyper-parameter β=1 for the Cora dataset and β=0.75 for the Citeseer dataset. As shown in Table 4.2, we obtain similar results on the Citeseer dataset with 7.51%(MLP) and 7.82%(SVC) accuracy improvements. Because GAT uses weighting neighbor features with feature-dependent and structure-free normalization, which does not rely on knowing the entire graph structure in advance, the local client can generate favorable embedding by partial information of a denser Cora dataset. Thus, our method is subject to further refining the embedding in this case. In addition to cleaning the Cora dataset, we randomly modify 15% labels as noisy (incorrect) labels during the training, leading to a considerable performance loss for clients such as Client3 and Client4. However, our proposed method can effectively mitigate the influence of noisy labels by integrating information from other clients. Consequently, the poor performance of Client3 or Client4 receives a significant improvement.

4.3.4 Impact of Alignment on Performance

To demonstrate the effectiveness of applying alignment during the FL aggregation, we plot the alignment precision of the public latent representations and the SVC classification accuracy of the graph embeddings corresponding to each training iteration of both frameworks in Figure 4.3 to Figure 4.6. We use k-nearest neighbors with k =1, 5, and 10 to measure the

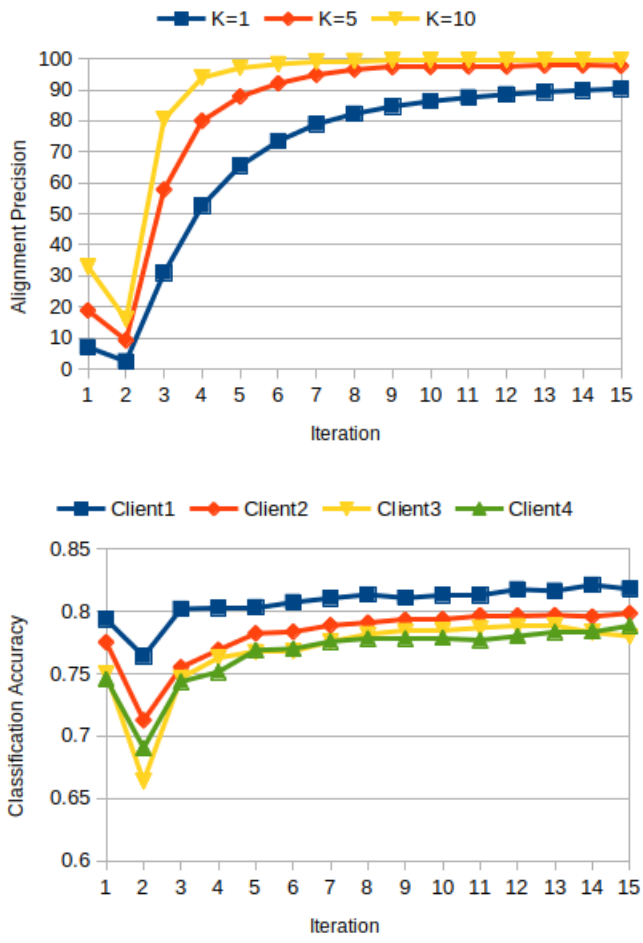


Figure 4.3 The KNN alignment precision and SVC classification accuracy corresponding to each iteration of the Federated DeepWalk Framework on the Cora dataset.

alignment precision between any pair of the public latent representations. Because we need to align each local representation to the dimension of the other clients, there are 12 pairs in the four clients' settings, and we only show the average value of 12 alignments in the figures as the variance is slight.

For the Federated DeepWalk framework in Figure 4.3 and Figure 4.4, although the classification accuracy of locally trained graph embedding is acceptable in the initial iteration, their

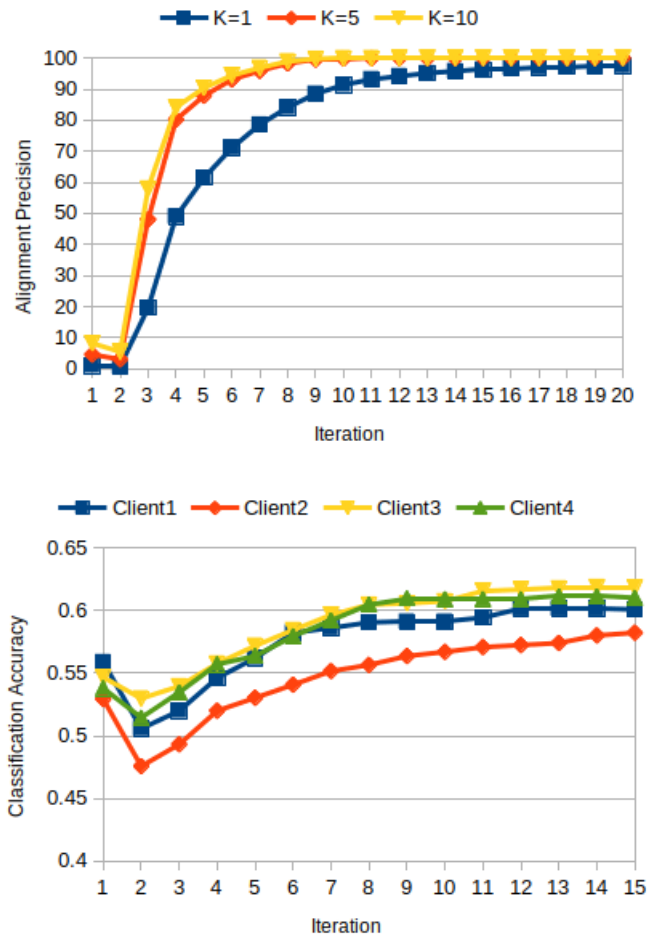


Figure 4.4 The KNN alignment precision and SVC classification accuracy corresponding to each iteration of the Federated DeepWalk Framework on the Citeseer dataset.

alignment results are inferior because of the random initialization. Consequently, we cannot integrate the information of different clients effectively, which leads to the performance diving in the second iteration. However, the rough integration in the first two iterations helps in the united initialization by setting the tone for the subsequent training. Thus, we observe that the quality of graph embedding improves with the promotion of the alignment effect, which can achieve above 90% precision of $k = 1$ at the convergence stage. For the

Federated GAT framework in Figure 4.5 and Figure 4.6, the initial representation alignment results are satisfactory with a fair classification accuracy of graph embedding. Moreover, we observe both alignment precision and classification accuracy surge in the second iteration after the federated learning process. Nevertheless, as we only use cosine-embedding loss at the intermediate layer, partial integrated information is squeezed out when the federated procedure converges within ten iterations. In general, there is a positive correlation between the alignment precision and classification accuracy, which confirms the effectiveness of our method.

4.4 Discussion

Through previous experiments, we find that our method performs better when applied to the Citeseer dataset, which is more sparse than the Cora dataset relatively. Because denser subgraphs mean the local clients have more information, limiting the improvement effect of federated learning. However, if the degrees of the shared nodes are low, they cannot comprehensively transmit the local information during the integration. Therefore, we design the supplemental experiments to further study the suitable application scenarios. Instead of randomly generating the subgraphs and selecting 40% public nodes to share, we compose the subgraphs with different percentages of top high-degree nodes from the original graph as the public nodes. We conduct the same embedding classification experiment and render the average accuracy of four clients in Table 4.3.

Under the DeepWalk framework, the classification accuracy of locally generated graph

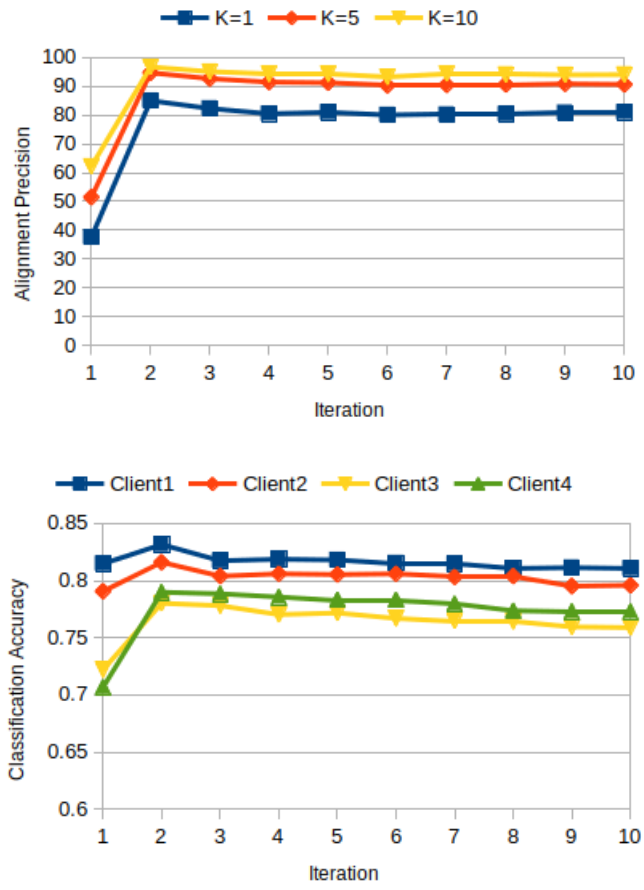


Figure 4.5 The KNN alignment precision and SVC classification accuracy corresponding to each iteration of the Federated GAT Framework on the Cora_Noisy dataset.

embeddings increases as the degree of nodes in the subgraph increases. Although federated learning can still improve the overall classification effect, the magnitude of improvement diminishes. With a simpler model DeepWalk, the local clients are more likely to get an underfit model with inferior prediction accuracy below 70%. Federated learning tackles the underfitting issue more by sharing public information between clients and indirectly increasing the local training dataset's size. In the experimental group of GAT, we notice that the higher subgraph density reduces the accuracy of local graph embedding. One

Percent	DeepWalk			GAT		
	LOC	FED	Diff	LOC	FED	Diff
5%	57.1	61.8	+4.7	74.4	75.1	+0.7
10%	57.3	62.3	+5.0	73.9	75.1	+1.2
20%	61.8	65.0	+3.2	71.8	73.5	+1.9
30%	63.4	65.8	+2.4	70.5	72.6	+2.1
40%	67.8	69.2	+1.4	70.5	73.0	+2.5

Table 4.3 Results of the different shared public nodes.

reason is that the subgraphs generated by our method are disassortative, and the local aggregation mechanism of GAT may fail on disassortative graphs, where nodes within local neighborhoods provide more noise than helpful feature information. Another reason is that the local model is overfitting the denser training subgraph. However, the federated learning setting prevents the local model from focusing on the training data, and the embedding alignment technique has regularization effects empirically to avoid overfitting. Overall, our approach is suitable for general application scenarios, and the improvement effect is more prominent when the local embedding effect is unsatisfied.

4.5 Conclusions

This study investigates the practical challenges associated with federated graph neural networks, particularly when dealing with non-IID datasets. We present a cutting-edge federated learning framework that employs embedding alignment to achieve a uniform latent representation across clients, facilitating effective information integration in a federated context. Our experimental results showcase the framework's ability to outperform local training in terms of data usability while still ensuring privacy. Future work might benefit from examining more sophisticated embedding alignment techniques for even more precise information

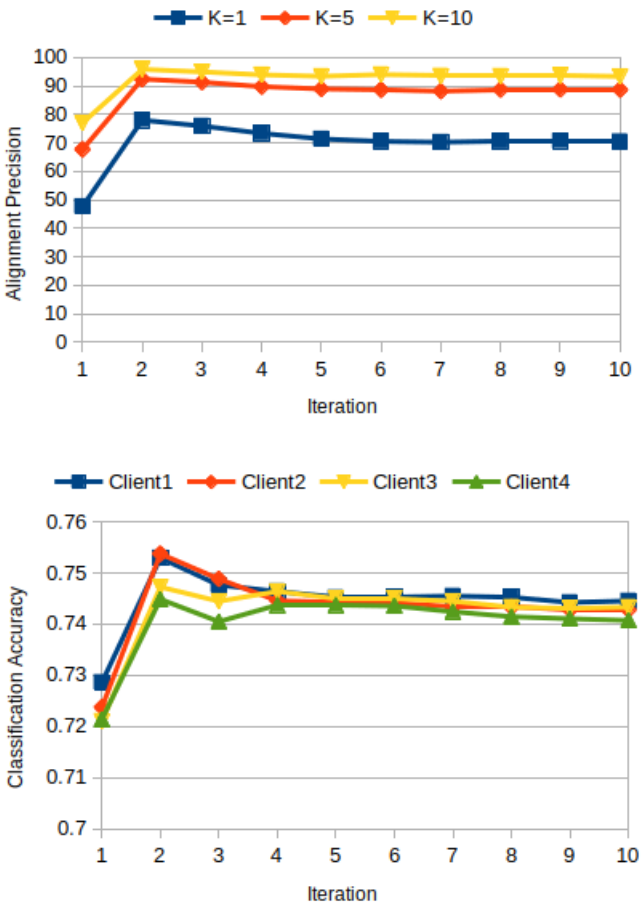


Figure 4.6 The KNN alignment precision and SVC classification accuracy corresponding to each iteration of the Federated GAT Framework on the Citeseer dataset.

integration. The study of shared public nodes remains a valuable avenue. As we look ahead, determining an optimal selection of public nodes to minimize sharing could offer a more nuanced equilibrium between privacy and data accessibility.

CHAPTER 5

Few-Shot Graph Classification with Structural-Enhanced Contrastive Learning

5.1 Introduction

The surge of big data has propelled machine learning to unprecedented heights. However, the efficacy of machine learning is often constrained by the quantity of input samples [112]. Drawing inspiration from the remarkable cognitive speed of humans, the concept of Few-Shot Learning (FSL) has emerged. As the name suggests, FSL seeks to learn from a limited set of samples. Distinct from traditional supervised learning, FSL, in the face of limited training data, aspires to learn a similarity metric to discern samples rather than classify them directly, enabling a wider application range.

The success of FSL in image processing has encouraged its extension into diverse areas, including graph analytics. While humans can readily differentiate between images, discerning complex graphs remains challenging, underscoring the importance of graph learning. Graph Neural Networks (GNNs), potent tools in the machine learning arsenal, leverage both graph structure and node information to execute an array of graph analytics tasks. These tasks span from edge-level to node-level and graph-level [98]. In particular, graph classification aligns with GNNs producing graph-level outputs, typically involving pooling and readout operations. The pooling layer [113, 114] condenses the graph, ensuring node representations on this compacted graph embody broader graph-level insights. Subsequently, the readout layer aggregates the hidden representations of these subgraphs to attain a succinct graph representation, which is then employed as the classification label for the entire graph.

Recently, researchers [115] compare GNNs methods for graph classification in a standardized and uniform evaluation framework. The findings highlight the potential underutilization of structural information, especially in chemical and social datasets. Two probable explanations surface: either an effective task solution can circumvent topological information, or GNNs must harness the graph structure more effectively. Given that structural features are intricately tied to molecular properties in chemistry, if endowed with ample molecular data (meaning large node-attribute dimensions), standard binary classification tasks (e.g., categorizing chemical compounds as active or inactive) might bypass the need for GNNs. Such an approach also clashes with FSL, given its intrinsic novelty of classes. Hence, our investigation gravitates towards the FSL challenges in social graphs, characterized by their intricate topologies and ambiguous relationships with node attributes. Figure 5.1 suggests that a mere focus on node attributes might impede classification accuracy without the appropriate structural context. A real-world implication of this issue manifests in challenges like the "cold start" problem seen in recommendation systems due to limited sample sizes [116].

Introducing a novel learning framework, SE-GCL, we aim to enhance the capabilities of GNNs in leveraging graph structure and addressing the challenges mentioned above. By utilizing meta-learning and contrastive learning techniques, our proposed framework achieves accurate graph classification results in an end-to-end process. The contributions of our work can be summarized as follows:

- We investigate a general scenario for few-shot graph classification tasks and present a

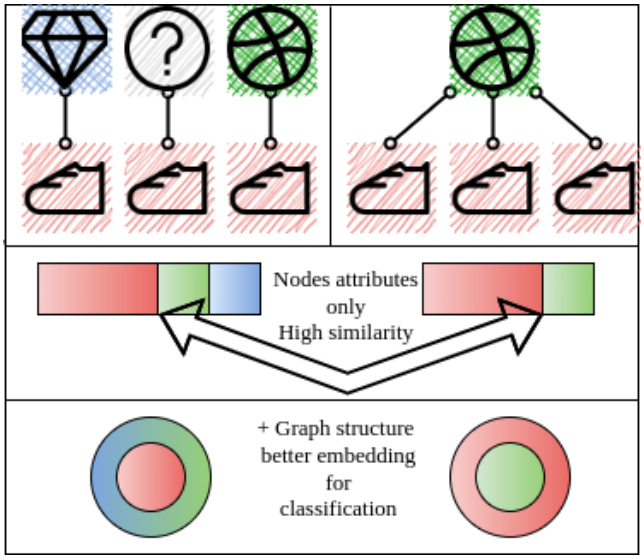


Figure 5.1 A simple example that shows node attributes may hurt classification accuracy without adequately considering the graph structure.

learning framework that integrates meta-learning and contrastive learning techniques. This integration allows us to achieve accurate graph classification results while ensuring the protection of data copyright.

- We construct two benchmark graph datasets with large node-attribute dimensions, designed for multi-class classification tasks. These datasets serve as valuable resources for future research in the field of few-shot graph classification. They enable researchers to evaluate and compare various algorithms and techniques within a context that respects data copyright protection.

- Through extensive experiments on ground truth datasets, we demonstrate the effectiveness of using contrastive learning techniques to enhance the utilization of graph topological information. Our framework achieves competitive performance when compared to other state-of-the-art methods.

5.2 Proposed Work

5.2.1 Problem Definition

FSL is usually applied in supervised learning for the task of object classification, also considered as N-way-K-shot classification. During the few-shot training phase, N categories (ways) with K samples (shots) per category are constructed as the support set first. Second, another batch of samples in N categories, named query set, is selected from the remaining data as the model's prediction object. Then the task is to distinguish these query set samples from the N*K support set.

We formulate our few-shot graph classification problem as a standard N-way-K-shot classification task, where a set of graph $\{G_1, G_2, ..., G_m\}$ and their labels $\{y_1, y_2, ..., y_m\}$ are given. Let $G = (\mathbf{U}, \mathbf{E}, \mathbf{A}, y)$ denote an undirected unweighted graph, where $\mathbf{U}$ is the set of nodes, $\mathbf{E}$ is the set of edges, $\mathbf{A}$ is the set of node-attributes, and y is the label associated with each graph. According to label y, $\{G_1, G_2, ..., G_m\}$ is split into $\{(G_{train}, y_{train})\}$ and $\{(G_{test}, y_{test})\}$ as the training set and test set respectively. Notice that y_{train} and y_{test} must have no common classes for the Meta-learning setting. In the Meta-training phase, we construct the support dataset $\mathbf{D}_S(G_{train}, y_{train})$ by randomly selecting K samples from each of the N classes and the query dataset $\mathbf{D}_Q(G_{train}, y_{train})$ containing other M samples from the same N classes. The goal is to predict the label of each graph in the query dataset by giving a limited number of support graphs (*i.e.*, N << M). At the Meta-testing stage, the same classification task is performed on the $\mathbf{D}_S(G_{test}, y_{test})$ and $\mathbf{D}_Q(G_{test}, y_{test})$ dataset with the disjoint label y_{test}, which verifies the result of knowledge transfer and adaptation.

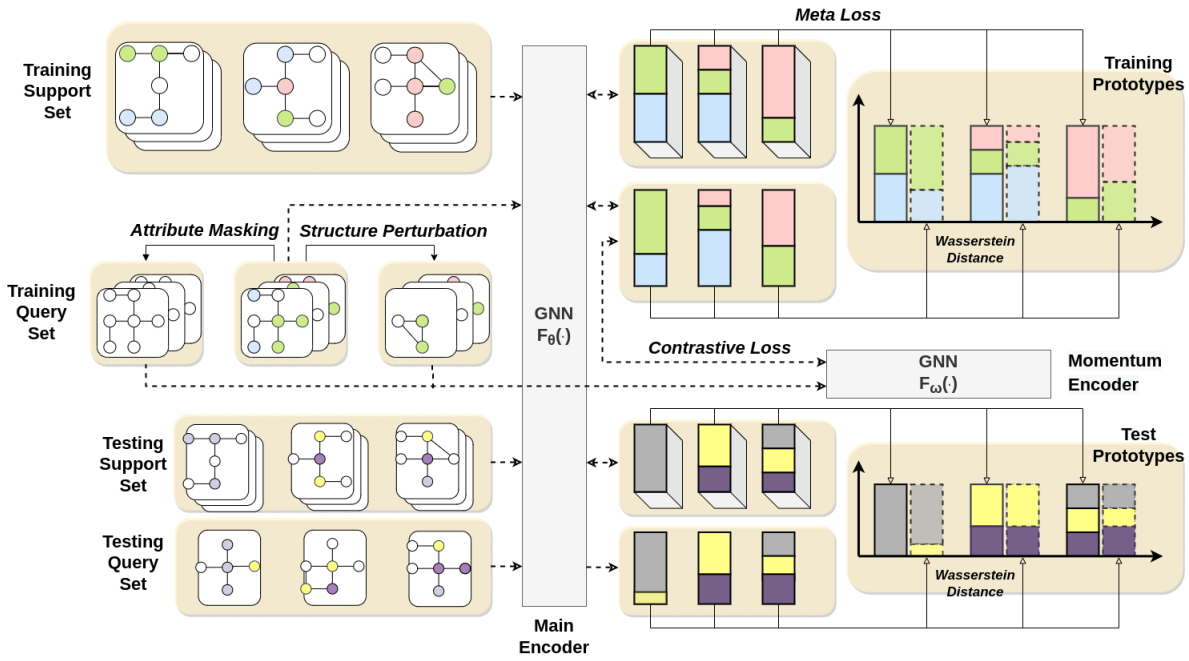


Figure 5.2 The overview of SE-GCL. The framework consists of two main processes: graph Meta-learning and contrastive learning. Given a support set of input graphs, we use a graph encoder to extract robust feature representation and derive reliable prototypes for each class. The Wasserstein metric measures the similarity between the query graph and the prototype. Further, we impose the contrastive loss on the query set to improve the model's generalizability. The complete workflow of all modules is an end-to-end solution. More details could be in the section of the proposed framework.

5.2.2 Proposed Framework

Figure 5.2 illustrates the framework of our proposed method. Two complementary classification tasks are performed simultaneously to learn the main encoder $\mathcal{F}_\theta(\cdot)$, which is a GNN for projecting a graph into an embedding. The first learning module is metric-based Meta-learning, which utilizes explicit label information to generate the graph embedding and compute the similarity between the support set and query set. The second learning module is contrastive learning, which is a self-supervised instance-level classification task to improve the representation result. For self-supervised learning, we design a strategy to generate a

pair of positive and negative augmentation views of the input graph automatically.

During Meta-learning, the main encoder $\mathcal{F}_\theta(\cdot)$ maps each graph into a latent representation as its graph embedding $\mathbf{h}_{G_i} = \mathcal{F}_\theta(G_i)$. Specifically, GNNs compute graph embedding via a message-passing framework:

$$\mathbf{h}_u^{(l+1)} = \mathrm{COM}\big(\mathbf{h}_u^{(l)}, \big[\mathrm{AGG}\big(\big\{\mathbf{h}_{u'}^{(l)} | \forall u' \in \mathbf{U}'\big\}\big)\big]\big), \tag{5.1}$$

$$\mathbf{h}_G^{(l)} = \mathrm{READOUT}\big(\mathbf{h}_u^{(l)} | \forall u \in \mathbf{U}\big), \tag{5.2}$$

where $\mathbf{h}_u^{(l)}$ denotes the embedding of node u at l-th GNN layer; $\mathbf{U}'$ is the neighbor set of node u; $\mathrm{AGG}(\cdot)$ is neighbor aggregation function; $\mathrm{COM}(\cdot)$ is combination function; $\mathrm{READOUT}(\cdot)$ is graph-level pooling function. Then all support graph embedding in the same class y_n aggregate into one prototype representation $\mathbf{z}_n$ by computing the average, which is formulated as:

$$\mathbf{z}_n = \frac{1}{K} \sum_{i=1}^{K} \mathbf{h}_{G_i} \, (G_i \in \mathbf{D}_S(G, y_n), n \in [1, N]), \tag{5.3}$$

To predict the label of the query graph, the similarity between query graph embedding and the prototype representation is measured by the p-th Wasserstein distance following [117], which is the optional cost of moving mass between two graph embeddings. The classification loss $\mathcal{L}_{Meta}$ is defined as the average cross entropy between true labels and predictions based on the similarity, which can be formulated as:

$$\mathcal{L}_{Meta}(\mathbf{D}_S, \mathbf{D}_Q, \theta) = -\frac{1}{M} \sum_{(G,y)\in\mathbf{D}_Q} log \frac{e^{sim(\mathcal{F}_\theta(G), z_y)}}{\sum_{i=1}^{N} e^{sim(\mathcal{F}_\theta(G), z_i)}}, \tag{5.4}$$

where sim denotes the Wasserstein similarity metric.

Because contrastive learning can maximize the agreement between the input data and its positive view while minimizing the agreement with the negative view, two augmentation operations are employed to generate a pair of differentiable views for the respective goals. Expressly, the positive augmentation operation preserves the original topology of the sample graph G_i and masks all the node features to form a positive view G_i^{mask}, which aims to mediate the overwhelming of the node features over the graph structure information in the representation learning. On the other hand, the negative augmentation operation generates a negative view G_i^{neg} by random node dropping and edge perturbation. Both operations follow an i.i.d. uniform distribution with node-dropping ratio η and edge-perturbation ratio $1-\eta$. For edge-perturbation, it randomly drops $1-\eta$ existing edges, then adds the same amount of random edges back into G_i. To form G_i^{neg} as a small subgraph from G_i with a few noisy edges, η is set at 0.8 by default. Moreover, the paper [72] states that the structural information of graph data consists of both local and global dimensions, which means some attributes of a graph depend on the substructure of the graph while some consider the global structure more. As generalization is the main challenge for Meta-learning to test novel domains, randomly treating a small subgraph as the negative example helps predictive models generalize beyond the limited training data. It should be noted that the negative view of one sample graph is also treated as the negative view of the rest samples (*i.e.*, for a query set containing M

samples, there are M negative views for each sample graph). Introduced in this paper [118], we apply a momentum encoder $\mathcal{F}_\omega(\cdot)$ for projecting the contrastive views, which behaves similarly as the main encoder as its parameter ω is a moving average of θ. Given $\mathcal{F}_\theta(G_i)$, the contrastive loss aims to maximize its agreement with $\mathcal{F}_\omega(G_i^{mask})$ while minimizing the agreement with all the negative views $\mathcal{F}_\omega(G_j^{neg}), j \in M$, which can be formulated as:

$$\begin{aligned} &\mathcal{L}_{con}(\mathbf{D}_S, \mathbf{D}_Q, \theta, \omega, \eta) \\ &= -\frac{1}{M} \sum_{G \in \mathbf{D}_Q} log \frac{e^{sim(\mathcal{F}_\theta(G), \mathcal{F}_\omega(G^{mask}))}}{\sum_{j=1}^{M} e^{sim(\mathcal{F}_\theta(G), \mathcal{F}_\omega(G_j^{neg}))}}, \end{aligned} \tag{5.5}$$

where M denotes the size of the query set, and η is the perturbation ratio. By minimizing $\mathcal{L}_{con}$ w.r.t θ, we force the main encoder $\mathcal{F}_\theta(\cdot)$ to maintain the complete structural information in the embedding and produce more generalized prototypical networks. Thus, the overall loss is the combination of the classification loss and the contrastive loss:

$$\mathcal{L}_{total} = \mathcal{L}_{Meta}(\mathbf{D}_S, \mathbf{D}_Q, \theta) + \beta \mathcal{L}_{con}(\mathbf{D}_S, \mathbf{D}_Q, \theta, \omega, \eta), \tag{5.6}$$

where β is a hyper-parameter that balances two terms. The detailed learning process is described in Algorithm 5.

5.3 Experiments

In this section, we present the experiments developed by PyTorch Geometric and conducted on a workstation with an Intel Core i7 2.80GHz CPU and an NVIDIA GeForce GTX 1070 GPU. Our proposed method is evaluated on standard few-shot learning benchmarks with real-world datasets. We also conduct an ablative study about the effectiveness of the con-

Algorithm 5 Learning process of SE-GCL

Input: Graph dataset: $\{G_1, G_2, ..., G_m\}$,
Graphs' labels: $\{y_1, y_2, ..., y_m\}$,
Task: $\mathcal{T}_{test}$=$\{\mathbf{D}_S(G_{test}, y_{test}), \mathbf{D}_Q(G_{test}, y_{test})\}$,
Training episodes: T, Perturbation ratio: η,
Learning rate: α, Momentum coefficient: ϵ.
Output: : Predicted labels of G_{test} in $\mathbf{D}_Q$

while $i < T$ **do**
  //Meta-training process
  Sample a Meta-training task:
    $\mathcal{T}^i_{train}$=$\{\mathbf{D}_S(G_{train}, y_{train}), \mathbf{D}_Q(G_{train}, y_{train})\}$;
  Compute the prototype representations $\mathbf{z}_{train}$ of support set $\mathbf{D}_S(G_{train})$ according to Eq. 5.3;
  //contrastive process
  Generate the augmentation views G^{mask} and G^{neg} of $\mathbf{D}_Q(G_{train})$ with η;
  Update the main encoder by minimizing loss in Eq. 5.6:
    $\theta^{i+1} = \theta^i - \alpha\nabla_{\theta^i}\mathcal{L}_{total}$;
  Update the momentum encoder with ϵ:
    $\omega^{i+1} = \epsilon\omega^i + (1-\epsilon)\theta^{i+1}$;
  //Meta-testing process
  Compute the prototype representations $\mathbf{z}_{test}$ of support set $\mathbf{D}_S(G_{test})$ from $\mathcal{T}_{test}$ according to Eq. 5.3;
  Predict the labels of $\mathbf{D}_Q(G_{test})$ from $\mathcal{T}_{test}$ using the prototypical networks.
end while
return Predicted labels of G_{test} in $\mathbf{D}_Q$

trastive learning module in our framework.

5.3.1 Datasets

In the experiments, we use a variety of large and small attributed networks that are collected from different domains, including e-commerce networks, citation networks, and morphological networks. It is worth noting that meta-learning tasks often demand a considerable number of classes, whereas some commonly employed graph datasets [119] have limited classes or small node-attribute dimensions. Therefore, to validate the effectiveness of the

Symbol	Description
G	undirected unweighted graph
$\mathbf{U}$	set of nodes
$\mathbf{U}'$	set of node's neighbors
$\mathbf{E}$	set of edges
$\mathbf{A}$	set of node attributes
y	graph label
$\mathbf{D}_S$	support dataset
$\mathbf{D}_Q$	query dataset
$\mathcal{F}_\theta(\cdot)$	main graph encoder
$\mathcal{F}_\omega(\cdot)$	momentum graph encoder
$\mathbf{h}_G$	graph embedding
$\mathbf{h}_u^{(l+1)}$	node embedding at l-th GNN layer
$\mathbf{z}_n$	graph prototype representation
G^{mask}	graph positive augmentation view
G^{neg}	graph negative augmentation view
η	perturbation ratio of G^{neg}
β	regularization hyperparameter

Table 5.1 List of notations used in this work.

proposed framework, we construct graph datasets from two large attributed networks with lots of node classes, Amazon-Clothing and DBLP, and use the node-label distribution as the label of the new graph. Meanwhile, we adapt two small attributed networks Letter-High and TRIANGLES, which are used in GSM [71] for a fair comparison. To ensure that appropriate data usage agreements are in place with the data owners or providers, we specify the scope and limitations of data usage. The detailed descriptions of these datasets are as follows:

Datasets	$\lvert G\rvert$	$Avg.\lvert U\rvert$	$Avg.\lvert E\rvert$	$\lvert A\rvert$	$\frac{\lvert y_{train}\rvert}{\lvert y_{test}\rvert}$
Amazon-Clothing	2000	32.15	192.50	9034	10/10
DBLP	2000	47.25	318.45	7202	10/10
Letter-High	2250	4.67	4.50	2	11/4
TRIANGLES	2000	20.85	35.50	1	7/3

Table 5.2 Statistics of the datasets. We show each dataset with the number of graphs $|G|$, the average number of nodes $Avg.|U|$, the average number of edges $Avg.|E|$, the dimensions of node attributes $|A|$, and the number of classes for training over testing $|y_{train}|/|y_{test}|$.

Amazon-Clothing. The dataset was originally collected by [120] and has been pre-

processed by [115] for the FSL study. In our constructed dataset, each graph represents a customer's shopping history, where each node corresponds to a product, and different products are connected if the same customer browses them. The product descriptions are used as node attributes. We customize 2000 graphs with 20 types of shopping habits from 77 kinds of products for FSL.

DBLP. The citation network is extracted from [121] with node features generated by [122] using the Bag-of-Words model. For this dataset, we follow the same construction method to customize 2000 graphs with 20 graph classes, where each node represents a paper and edges represent citations.

Letter-High. Each graph is a distorted alphabetic prototype graph with undirected edges and vertices representing lines and ending points of lines [123]. More specifically, Letter-High contains 15 categories from the English alphabet: A, E, F, H, I, K, L, M, N, T, V, W, X, Y, and Z.

TRIANGLES. This dataset contains 10 different graph classes numbered from 1 to 10, corresponding to the number of triangles in the graphs of each class. The partial version is used in the experiments in [71] that reduces the graph sample size from 45,000 to 2,000.

5.3.2 Baselines and Implementation

We compare our method with the following five types of baselines:

Weisfeiler-Lehman Graph Kernels [124], based on the Weisfeiler-Lehman (WL) test of graph isomorphism, is considered as the state-of-the-art in graph classification. We skip the unsuitable Meta-training phase for this method and perform N-way-K-shot graph

Methods	Amazon-Clothing		
	5-way 5-shot	5-way 10-shot	8-way 5-shot
WL kernel	56.40 ± 2.23	65.24 ± 1.37	49.47 ± 2.64
GIN	63.25 ± 1.63	71.24 ± 1.57	55.47 ± 3.34
MAML (GCN)	70.72 ± 3.88	76.62 ± 2.35	60.70 ± 4.53
MAML (GAT)	70.66 ± 3.53	76.68 ± 2.51	60.27 ± 4.49
PN (GCN)	70.18 ± 1.19	77.43 ± 1.87	63.17 ± 2.14
PN (GAT)	71.22 ± 2.43	77.06 ± 2.15	63.89 ± 2.94
SE-GCL (GCN)	74.98 ± 2.01	80.22 ± 1.55	66.37 ± 1.99
SE-GCL (GAT)	**75.02 ± 2.90**	**81.76 ± 2.36**	**66.92 ± 2.43**

Methods	DBLP		
	5-way 5-shot	5-way 10-shot	8-way 5-shot
WL kernel	57.12 ± 2.44	65.52 ± 1.71	50.35 ± 2.39
GIN	66.10 ± 2.41	72.38 ± 1.44	57.13 ± 2.88
MAML (GCN)	73.12 ± 4.65	77.69 ± 2.89	63.19 ± 5.12
MAML (GAT)	74.10 ± 4.19	78.03 ± 3.44	62.80 ± 3.99
PN (GCN)	74.32 ± 2.49	79.79 ± 2.19	64.49 ± 3.19
PN (GAT)	74.91 ± 3.29	80.29 ± 2.34	64.52 ± 3.52
SE-GCL (GCN)	77.31 ± 2.17	83.40 ± 1.14	67.59 ± 2.86
SE-GCL (GAT)	**78.16 ± 3.09**	**84.75 ± 1.82**	**68.25 ± 3.20**

Table 5.3 Accuracy with a standard deviation of baselines and our method. We tested 100 N-way-K-shot tasks on both Amazon-Clothing and DBLP datasets. The best results are highlighted in bold.

Methods	K-shot	Letter-High	TRIANGLES
GSM	5	69.91 ± 5.90	71.40 ± 4.34
	10	73.28 ± 3.46	75.60 ± 3.67
	20	77.38 ± 1.58	80.04 ± 2.20
SE-GCL	5	74.34 ± 1.03	77.36 ± 1.25
	10	79.42 ± 0.84	83.14 ± 1.07
	20	84.15 ± 0.77	89.17 ± 0.85

Table 5.4 Accuracy of GSM and our method. We tested 100 N-way-K-shot tasks on both Letter-High (4-way) and TRIANGLES datasets (3-way).

classification directly on the testing dataset.

GIN [73] uses injective neighbor aggregation to approximately conceive through WL test, which considers performing better than GCN and GraphSAGE by [125] in case of graph classification. Thus, we train a naive GIN + MLP classifier directly on the testing dataset to verify the knowledge transfer ability in Meta-learning.

MAML [69] is an optimization-based Meta-learning method that tries to learn better model initialization from a series of Meta-training tasks. For the few-shot graph classification task, we extend it by the same graph encoder backbone in our framework.

GSM [71] clusters the graph classes based on the graph spectral measures into groups named super-classes and uses the constructed super-classes for few-shot learning.

PN has the identical architecture of our framework but without the contrastive learning module, which can be considered a variant of a Prototypical Network.

For either baselines or our framework, we implement the graph encoder consisting of three GCN layers [126] or GAT layers [127] with dimension sizes 32, 32, and 16, respectively. All the layers are activated with the ReLU function. We choose MinCUT Pooling from [128] as the readout operation because it coarsens a graph by taking into account both the connectivity structure and the node features. The loss is trained with Adam optimizer, whose learning rate is set to 0.001 initially with a weight decay of 0.0001. We adjust the dropout rate and the perturbation ratio η for each dataset to achieve the best performance and train the model with an early-stopping strategy across 300 episodes. Moreover, we set the regularization hyperparameter β to 1.0 and the momentum coefficient ϵ to 0.99 by default.

5.3.3 Result Analysis

We evaluate the performance by 5-way-5-shot, 5-way-10-shot, and 8-way-5-shot tasks on both Amazon-Clothing and DBLP datasets, whose metrics Accuracy (ACC) results are presented in Table 5.3. For Letter-High and TRIANGLES containing few node information, we perform

4-way and 3-way tasks, respectively, and only compare them with GSM's best results in Table 5.4. From a comprehensive view, we have the following observations:

• For WL kernel and GIN baselines, we perform N-way-K-shot graph classification directly over the test classes. It is clear that the sample size restricts the accuracy of the prediction results. When there are insufficient samples for training, the model is prone to overfit, which leads to unsatisfactory test results. GIN incorporates node features while generalizing the WL test, so the effect is better than the WL kernel, which shows that node features play a vital role in graph learning.

• Both MAML and PN achieve superior performances on three types of graph classification tasks over GIN, indicating that for Amazon-Clothing and DBLP datasets, Meta-learning can improve the learning process of new tasks using the experience gained from solving predecessor problems. MAML and PN have similar performance on 5-way tasks, but PN obtains about 3.5% performance gains on 8-way tasks. The reason is twofold: as an optimization-based approach, the generalization ability of MAML is getting poor when the number of classification labels increases and the difference in sample data becomes large. The performance suffers from its fine-tuning process. As a metric-based approach, PN learns generalizable matching metrics by taking the mean vector of support examples, which is simple but stable with favorable distance metrics.

• It is worth noting that SE-GCL extends the PN basis with the contrastive learning method, further enhancing the model's generalization ability. Forged by contrastive learning, the positive and negative samples strengthen the graph structure learning and make up for

the insufficient number of few-shot samples to a certain extent. Overall, SE-GCL outperforms the baselines in all the tasks on both Amazon-Clothing and DBLP datasets. At the same time, we also find that as the encoder backbone, the gap between GCN and GAT is not apparent, which means that the aggregation function (*i.e.*, the main difference in Message Passing between GCN and GAT) has much less impact on the graph-level than the pooling and readout operations.

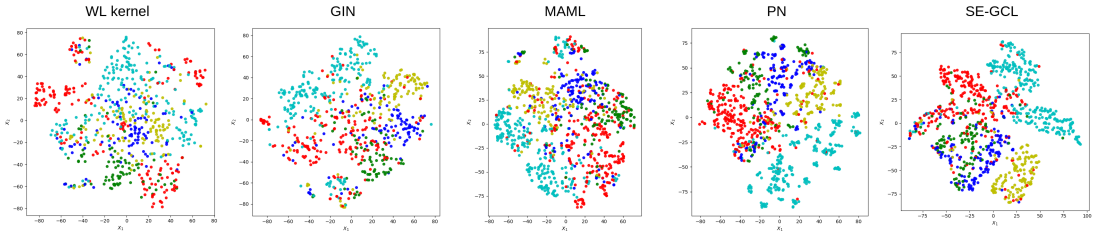


Figure 5.3 t-SNE visualization comparison for the DBLP dataset. The methods from left to right are WL kernel, GIN, MAML, PN, and our method (SE-GCL). Each class is represented in a different color.

- As shown in Table 5.3, SE-GCL outperforms GSM by 5% for both datasets, though both SE-GCL and GSM use metric-based approaches to solve the few-shot problem. This is because GSM assumes the test classes could belong to the same super-classes built from the training classes. However, training and test classes typically do not overlap in the few-shot setting. On the other hand, we believe the ability of the GNN encoder to learn graph representation in a top-down way is more critical when encountering unseen classes, where the effectiveness of the MinCUT Pooling strategy on unsupervised node clustering helps in the simple-graph dataset with few nodes information. Moreover, SE-GCL can alleviate the overfitting problem caused by simple graph topology through contrastive learning; even SE-GCL is more suitable for graph datasets with complex topology and excessive node

information. By incorporating such complexities into the graph data, it becomes more challenging for unauthorized individuals to extract or identify specific sensitive information from the copyrighted data.

• We visualize the DBLP dataset in 2-dimensional space by applying the t-SNE algorithm to the graph embeddings, which are learned from 5-way-10-shot classification tasks using the baselines and our method. The results shown in Figure 5.3 validate our assertions in the previous section through a meaningful layout. We can see that our method creates better clusters with low intra-cluster and high inter-cluster distances.

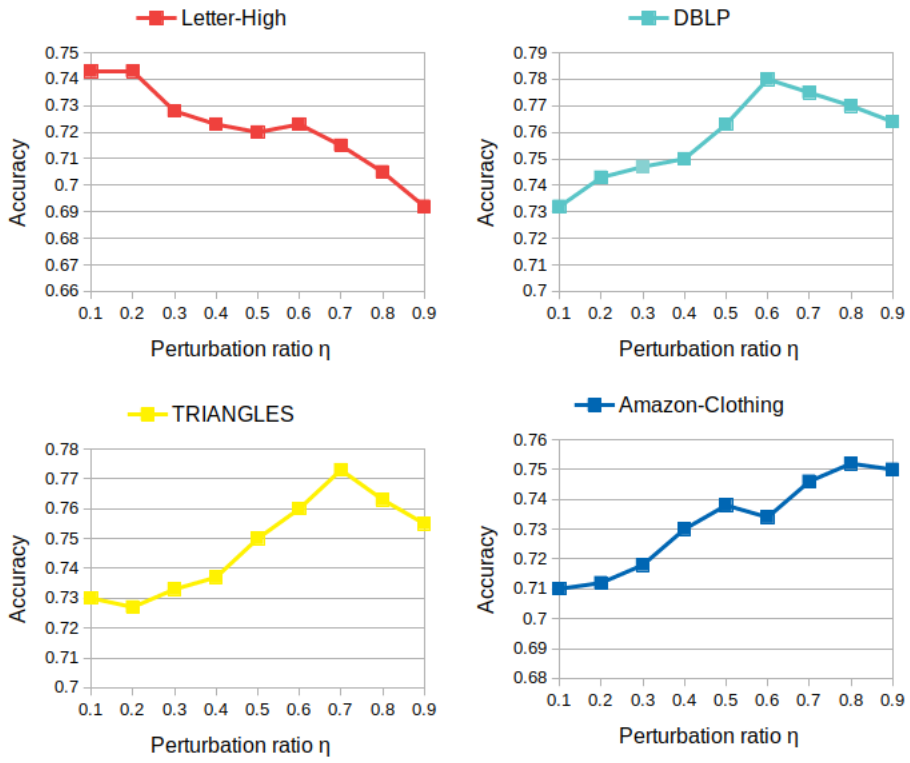


Figure 5.4 The influence of the perturbation ratio η. The range of η is set from 0.1 to 0.9.

5.3.4 Parameter Analysis

In this section, extensive experiments are conducted to analyze the influence of the perturbation ratio η to SE-GCL, whose results are shown in Figure 5.4. We review the negative augmentation operation as a combination of node-dropping and edge-perturbation with ratios η and 1-η, respectively. A small η renders a negative view having similar nodes in the sample graph but connected by different edges, while a hefty η generates a subgraph of the sample graph with a few random edges. According to Figure 5.4, the performances are better with large values of η in datasets DBLP, TRIANGLES, and Amazon-Clothing, while the performance deteriorates in dataset Letter-High with η increasing. There are two main reasons behind this phenomenon: First, the graphs of Letter-High are sparse with a moderately low number of nodes. With a large node-dropping ratio, the negative view may only have 1 or 2 nodes, which cannot provide valuable information for contrastive learning. Moreover, these small attributed graphs are more sensitive to individual edges, leading edge-perturbation to generate meaningful negative views. Second, the graphs of DBLP, TRIANGLES, and Amazon-Clothing have a large number of nodes with complex topology. Negative views generated by extensive edge-perturbation disturb the contrastive module due to being incompatible with node attributes and are empirically unhelpful for downstream performance. In contrast, node-dropping and subgraphs are beneficial across datasets by enforcing the consistency and generality of local sample graphs and global prototypes.

5.4 Conclusions

In this study, we address a practical issue in few-shot graph classification. We observe an imbalance in the utilization of node attributes and graph structure during the learning process and propose a novel Meta-learning framework with a contrastive learning module to enhance the learning of graph structure. On the one hand, the prototype networks based on the Wasserstein similarity metric allow the uncertainty distribution to encompass task embeddings beyond the training set, which enables the model to generalize to unseen test tasks after Meta-training. On the other hand, the contrastive module introduces meaningful positive and negative views, which regularize the model to prioritize the global structure of the graph over partial node attributes or subgraph features. The experimental results demonstrate that our framework achieves outstanding performance compared to other baselines, whether applied to large or small attributed graph datasets. As a future direction, we aim to develop automatic augmentation strategies within the contrastive learning module to prevent unauthorized use of original works and copyright infringement. By defining the objectives and parameters of data augmentation, organizations can exercise control over synthetic data to ensure compliance with copyright and privacy regulations.

CHAPTER 6
CONCLUSIONS

This dissertation delves deeply into the intricate world of Graph Neural Networks (GNNs), highlighting the multifaceted interplay between data privacy and utility. Through meticulous exploration of social graph embedding, non-IID challenges in federated learning, and the subtleties of few-shot graph classification, we unearth pivotal insights that expand the contemporary understanding of GNNs and suggest revolutionary shifts in prevailing methodologies.

In our initial inquiry into social graph embedding, we tackle an emerging privacy challenge and introduce a novel framework that harmonizes graph auto-encoders and adversarial learning. This synergy allows for nuanced latent representation regularization and ensures the effective removal of sensitive data. Empirical evaluations validate the superiority of our model over conventional methods, both in privacy preservation and data usability. As the horizon of GNNs widens, the burgeoning field of dynamic graph embedding surfaces as a critical area, especially given the potential risks associated with adversaries exploiting graph dynamism to extract sensitive information.

Progressing further, we confront the intricacies of federated graph neural networks, especially with non-IID datasets. Our innovative federated learning framework, anchored by embedding alignment, emerges as a beacon. It fosters consistent latent representation among clients, thereby enhancing federated information synthesis. The empirical analyses reflect the prowess of this framework over localized training, balancing data usability and privacy.

The future holds promise in refining embedding alignment technologies, and further improving information integration accuracy. Delving deeper into shared public nodes also presents a valuable avenue, aiming for an optimal node composition that harmonizes data availability with privacy.

Our final focus revolves around few-shot graph classification and addresses the evident disproportion in the use of node attributes versus graph structures. Our groundbreaking Meta-learning framework, augmented with a contrastive learning component, not only broadens the model's adaptability to new test scenarios but also underscores the significance of holistic graph structures. This framework consistently demonstrates superior performance metrics and overshadows standard baseline models across a spectrum of graph datasets. Envisaging the future, integrating automated augmentation strategies within our contrastive learning approach emerges as a pivotal step. Beyond mere performance optimization, this also serves as a protection against potential copyright breaches, empowering organizations to more stringently oversee data augmentation processes, and ensuring ethical and regulatory alignment.

Despite the profound insights yielded, it's imperative to acknowledge inherent research limitations. The datasets might not reflect the comprehensive breadth of real-world scenarios, which could affect on broader applicability. Our methods, processed under specific test conditions, may vary in different settings. Evolving technological landscapes and third-party dependencies could pose challenges for future replications and relevance. Additionally, potential biases in qualitative elements and potential shortcomings in our chosen evaluation

metrics necessitate careful interpretation of findings.

As GNNs forge ahead, dynamic graph embedding stands out as a crucial area, especially with the challenges associated with safeguarding sensitive data from potential adversarial threats. Expanding upon our accomplishments in embedding alignment promises more refined methodologies in the future. Equally, the nuances of shared public nodes in federated contexts beckon deeper exploration. The ever-present risk of copyright infringements in AI propels the need for refined automated augmentation techniques, ensuring both model resilience and copyright protection. Given the successes of our few-shot graph classification framework, its potential adaptability across diverse graph datasets remains a promising venture. In summation, this dissertation, while offering a comprehensive examination of GNNs and charting new domains, emphasizes the dynamic, evolving landscape of graph learning, marked by emerging challenges and untapped opportunities.

Appendix of Chapter 3

Approaches	25%		50%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	82.2 ± 1.7	89.9 ± 4.6	82.0 ± 1.2	86.3 ± 3.4
GAE_RM	80.8 ± 1.0	82.1 ± 6.9	84.5 ± 1.3	81.3 ± 4.8
DPNE	54.6 ± 1.3	65.7 ± 6.1	55.0 ± 1.7	63.7 ± 8.1
NPGE	86.4 ± 1.4	85.7 ± 5.2	86.3 ± 1.0	82.6 ± 3.9
LPPGE(T)	81.9 ± 1.8	81.1 ± 4.3	82.4 ± 1.1	77.7 ± 3.9
LPPGE(A)	**80.7 ± 1.0**	**73.0 ± 5.1**	**80.7 ± 0.9**	**64.6 ± 4.7**
Approaches	75%		100%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	82.4 ± 1.8	85.1 ± 4.0	81.8 ± 1.0	85.6 ± 3.5
GAE_RM	85.0 ± 1.1	84.5 ± 4.1	84.6 ± 0.7	83.3 ± 3.0
DPNE	53.8 ± 1.7	65.9 ± 4.3	55.3 ± 1.3	67.3 ± 4.2
NPGE	84.3 ± 2.4	85.2 ± 3.4	85.1 ± 1.4	89.2 ± 2.8
LPPGE(T)	82.4 ± 1.3	78.1 ± 3.2	81.5 ± 1.1	75.8 ± 3.9
LPPGE(A)	**81.2 ± 1.2**	**72.6 ± 3.4**	**80.5 ± 1.2**	**72.3 ± 3.1**

Table A.1 Results of link prediction for the Cora dataset.

Approaches	25%		50%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	84.9 ± 1.2	94.3 ± 3.2	84.8 ± 2.6	93.7 ± 2.1
GAE_RM	85.0 ± 1.6	97.0 ± 4.4	86.0 ± 1.8	87.2 ± 6.0
DPNE	52.9 ± 1.7	64.0 ± 6.4	52.2 ± 1.5	62.2 ± 6.0
NPGE	87.5 ± 0.9	89.3 ± 6.5	86.9 ± 1.0	87.1 ± 4.4
LPPGE(T)	**87.9 ± 1.6**	**82.6 ± 6.1**	**85.8 ± 1.2**	**75.8 ± 4.9**
LPPGE(A)	84.8 ± 1.2	86.0 ± 6.7	82.1 ± 3.6	80.3 ± 4.5
Approaches	75%		100%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	85.0 ± 1.7	91.3 ± 3.6	83.7 ± 1.4	92.1 ± 3.4
GAE_RM	86.6 ± 0.9	92.6 ± 2.6	87.5 ± 1.2	91.8 ± 3.3
DPNE	53.4 ± 1.6	64.0 ± 4.4	52.7 ± 1.3	67.5 ± 5.2
NPGE	88.5 ± 0.6	88.7 ± 2.5	88.9 ± 1.5	92.6 ± 2.6
LPPGE(T)	84.0 ± 0.6	84.0 ± 4.3	85.6 ± 1.6	81.0 ± 3.9
LPPGE(A)	**83.9 ± 1.4**	**77.1 ± 3.7**	**84.1 ± 1.2**	**77.0 ± 3.5**

Table A.2 Results of link prediction for the Citeseer dataset.

Approaches	25%		50%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	84.6 ± 0.3	89.7 ± 0.7	84.5 ± 0.4	89.1 ± 1.2
GAE_RM	83.9 ± 0.3	88.1 ± 0.1	83.9 ± 0.3	86.9 ± 1.0
DPNE	39.9 ± 10.4	41.2 ± 12.9	45.7 ± 11.4	47.8 ± 13.5
NPGE	82.0 ± 0.3	85.8 ± 1.0	82.2 ± 0.3	86.4 ± 0.6
LPPGE(T)	81.5 ± 0.2	84.5 ± 0.5	80.9 ± 0.4	82.5 ± 0.1
LPPGE(A)	**80.1 ± 0.2**	**83.6 ± 0.9**	**80.2 ± 0.2**	**82.4 ± 1.0**
Approaches	75%		100%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	84.5 ± 0.5	89.9 ± 1.1	84.4 ± 1.4	90.0 ± 1.2
GAE_RM	84.1 ± 0.3	87.7 ± 0.5	84.2 ± 0.2	89.6 ± 0.8
DPNE	44.3 ± 12.4	46.0 ± 14.6	48.8 ± 10	52.5 ± 10.2
NPGE	82.3 ± 0.4	86.5 ± 0.5	81.7 ± 0.1	85.2 ± 0.6
LPPGE(T)	81.4 ± 0.2	82.8 ± 0.6	81.2 ± 0.2	83.1 ± 0.5
LPPGE(A)	**78.1 ± 0.2**	**78.4 ± 0.9**	**80.2 ± 0.2**	**80.7 ± 1.0**

Table A.3 Results of link prediction for the Yale dataset.

Approaches	25%		50%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	84.9 ± 0.2	89.7 ± 1.0	85.2 ± 0.3	90.3 ± 1.1
GAE_RM	84.7 ± 0.5	88.2 ± 1.6	84.8 ± 0.3	89.1 ± 1.2
DPNE	54.0 ± 12.7	65.2 ± 18.5	55.8 ± 11.9	67.9 ± 18.0
NPGE	83.5 ± 0.3	88.3 ± 1.1	83.4 ± 0.3	88.1 ± 0.7
LPPGE(T)	82.1 ± 0.3	81.9 ± 0.9	82.1 ± 0.2	84.4 ± 0.8
LPPGE(A)	**79.2 ± 0.1**	**75.4 ± 1.5**	**78.7 ± 0.5**	**76.3 ± 1.4**
Approaches	75%		100%	
	Non-sen	Sensitive	Non-sen	Sensitive
GAE	85.1 ± 0.3	90.1 ± 1.0	85.1 ± 0.4	90.7 ± 1.1
GAE_RM	84.6 ± 0.4	89.2 ± 1.3	84.7 ± 0.3	88.5 ± 0.7
DPNE	58.5 ± 8.9	71.2 ± 13.3	60.6 ± 1.2	75.9 ± 3.9
NPGE	83.1 ± 0.4	88.7 ± 1.1	83.5 ± 0.4	88.0 ± 1.2
LPPGE(T)	81.8 ± 0.4	83.2 ± 0.8	82.2 ± 0.3	83.1 ± 0.7
LPPGE(A)	**78.4 ± 0.5**	**75.1 ± 1.4**	**80.4 ± 0.3**	**81.3 ± 1.2**

Table A.4 Results of link prediction for the Rochester dataset.

Approaches	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	73.7	71.5	74.3	71.5	73.1	71.5	74.0	71.5
GAE_RM	72.7	70.1	78.2	73.6	76.3	75.1	75.1	71.3
DPNE	15.2	6.63	14.9	6.63	14.8	6.63	14.5	6.63
NPGE	69.3	60.2	73.0	61.6	73.9	63.9	72.4	68.0
LPPGE(T)	73.6	59.6	75.8	69.8	73.7	54.8	79.2	70.8
LPPGE(A)	72.0	66.9	71.0	61.7	70.7	62.1	73.9	71.8

Table A.5 Results of node classification for the Cora dataset (7 categories).

Approaches	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	59.2	54.1	61.2	54.0	58.9	54.1	58.6	54.0
GAE_RM	58.2	51.9	58.7	54.1	51.9	48.4	63.5	55.5
DPNE	16.7	8.32	17.5	8.32	17.3	8.32	18.6	8.32
NPGE	64.7	56.0	67.3	60.5	67.9	62.6	69.0	64.1
LPPGE(T)	64.1	55.0	66.5	60.7	64.6	57.3	56.4	47.7
LPPGE(A)	60.3	54.7	59.2	57.7	59.6	55.6	66.9	62.3

Table A.6 Results of node classification for the Citeseer dataset (6 categories).

Approaches	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	86.5	87.1	86.3	87.1	86.1	87.0	85.8	87.0
GAE_RM	85.2	86.7	85.1	85.8	84.3	86.9	85.1	86.3
DPNE	24.5	4.66	24.6	4.66	24.2	4.66	24.2	4.66
NPGE	83.1	81.4	83.9	82.5	82.8	82.0	78.9	77.6
LPPGE(T)	83.8	82.8	83.8	81.8	84.6	82.8	84.8	83.5
LPPGE(A)	85.4	84.7	83.9	83.0	82.9	81.2	84.0	83.3

Table A.7 Results of node classification for the Yale dataset (6 categories).

Approaches	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	84.6	84.1	85.0	84.1	84.6	84.1	84.4	84.1
GAE_RM	85.4	83.4	84.1	83.6	85.1	83.5	85.2	84.2
DPNE	50.4	44.7	50.8	44.7	49.7	44.7	50.2	44.7
NPGE	83.7	81.6	85.0	79.9	85.3	82.6	84.1	80.0
LPPGE(T)	85.4	81.5	85.0	80.9	85.8	82.9	86.2	82.8
LPPGE(A)	83.7	79.7	85.1	79.0	85.1	82.0	86.4	83.6

Table A.8 Results of node classification for the Rochester dataset (3 categories).

REFERENCES

[1] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: A survey toward private and secure applications," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–38, 2021.

[2] Z. Cai, X. Zheng, J. Wang, and Z. He, "Private data trading towards range counting queries in internet of things," *IEEE Transactions on Mobile Computing*, 2022.

[3] X. Zheng, L. Tian, G. Luo, and Z. Cai, "A collaborative mechanism for private data publication in smart cities," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7883–7891, 2020.

[4] Z. Xiong, W. Li, Q. Han, and Z. Cai, "Privacy-preserving auto-driving: a gan-based approach to protect vehicular camera data," in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 668–677, IEEE, 2019.

[5] Z. Xiong, Z. Cai, Q. Han, A. Alrawais, and W. Li, "Adgan: Protect your location privacy in camera data of auto-driving vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6200–6210, 2020.

[6] H. Xu, Z. Cai, and W. Li, "Privacy-preserving mechanisms for multi-label image recognition," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 4, pp. 1–21, 2022.

[7] H. Xu, Z. Cai, D. Takabi, and W. Li, "Audio-visual autoencoding for privacy-preserving video streaming," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1749–1761, 2021.

[8] J. Wang, Z. Cai, and J. Yu, "Achieving personalized k-anonymity-based content privacy for autonomous vehicles in cps," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4242–4251, 2019.

[9] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Network*, vol. 32, no. 4, pp. 8–14, 2018.

[10] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492–6499, 2019.

[11] H. Xu, W. Li, and Z. Cai, "Analysis on methods to effectively improve transfer learning performance," *Theoretical Computer Science*, vol. 940, pp. 90–107, 2023.

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

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

[14] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world*

wide web, pp. 1067–1077, 2015.

[16] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, 2016.

[17] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[18] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[19] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, pp. 593–607, Springer, 2018.

[20] K. Li, G. Lu, G. Luo, and Z. Cai, "Seed-free graph de-anonymiztiation with adversarial learning," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 745–754, 2020.

[21] S. Ruder, I. Vulić, and A. Søgaard, "A survey of cross-lingual word embedding models," *Journal of Artificial Intelligence Research*, vol. 65, pp. 569–631, 2019.

[22] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," in *International Conference on Learning Representations*, 2018.

[23] G. Dinu, A. Lazaridou, and M. Baroni, "Improving zero-shot learning by mitigating the hubness problem," *arXiv preprint arXiv:1412.6568*, 2014.

[24] Z. Sun, W. Hu, Q. Zhang, and Y. Qu, "Bootstrapping entity alignment with knowledge

graph embedding.," in *IJCAI*, vol. 18, pp. 4396–4402, 2018.

[25] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo, and Y. Qu, "Multi-view knowledge graph embedding for entity alignment," *arXiv preprint arXiv:1906.02390*, 2019.

[26] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[27] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5812–5823, 2020.

[28] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *International Conference on Machine Learning*, pp. 12121–12132, PMLR, 2021.

[29] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, pp. 2069–2080, 2021.

[30] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *arXiv preprint arXiv:2006.04131*, 2020.

[31] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "Infogcl: Information-aware graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30414–30425, 2021.

[32] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," *Advances in Neural Information Processing Systems*,

vol. 34, pp. 15920–15933, 2021.

[33] S. Lin, C. Liu, P. Zhou, Z.-Y. Hu, S. Wang, R. Zhao, Y. Zheng, L. Lin, E. Xing, and X. Liang, "Prototypical graph contrastive learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[34] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1294–1303, 2022.

[35] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[36] S. Zhu, W. Li, H. Li, L. Tian, G. Luo, and Z. Cai, "Coin hopping attack in blockchain-based iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4614–4626, 2018.

[37] S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li, "zkcrowd: a hybrid blockchain-based crowdsourcing platform," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4196–4205, 2019.

[38] C. Wang, Z. Cai, and Y. Li, "Sustainable blockchain-based digital twin management architecture for iot devices," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6535–6548, 2022.

[39] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in social networks," in *Proceedings of the 17th ACM conference on Information and knowledge management*,

pp. 289–298, 2008.

[40] X. Ying and X. Wu, "On link privacy in randomizing social networks," *Knowledge and information systems*, vol. 28, no. 3, pp. 645–663, 2011.

[41] M. Fire, G. Katz, L. Rokach, and Y. Elovici, "Links reconstruction attack," in *Security and Privacy in Social Networks*, pp. 181–196, Springer, 2013.

[42] A. M. Fard and K. Wang, "Neighborhood randomization for link privacy in social network analysis," *World Wide Web*, vol. 18, no. 1, pp. 9–32, 2015.

[43] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2016.

[44] Y. Huang, Y. J. Li, and Z. Cai, "Security and privacy in metaverse: A comprehensive survey," *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 234–247, 2023.

[45] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart internet of things systems: a consideration from a privacy perspective," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 55–61, 2018.

[46] D. Xu, S. Yuan, X. Wu, and H. Phan, "Dpne: Differentially private network embedding," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 235–246, Springer, 2018.

[47] S. Zhang and W. Ni, "Graph embedding matrix sharing with differential privacy," *IEEE Access*, vol. 7, pp. 89390–89399, 2019.

[48] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with

non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[49] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1698–1707, IEEE, 2020.

[50] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[51] X. Zhang, M. Hong, S. Dhople, W. Yin, and Y. Liu, "Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data," *arXiv preprint arXiv:2005.11418*, 2020.

[52] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," *arXiv preprint arXiv:2102.07623*, 2021.

[53] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 965–978, IEEE, 2022.

[54] Z. Xiong, Z. Cai, D. Takabi, and W. Li, "Privacy threat and defense for federated learning with non-iid data in aiot," *IEEE Transactions on Industrial Informatics*, 2021.

[55] C. Chi, Y. Wang, X. Tong, M. Siddula, and Z. Cai, "Game theory in internet of things: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12125–12146, 2021.

[56] G. Mei, Z. Guo, S. Liu, and L. Pan, "Sgnn: A graph neural network based federated learning approach by hiding structure," in *2019 IEEE International Conference on Big*

Data (Big Data), pp. 2560–2568, IEEE, 2019.

[57] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *arXiv preprint arXiv:1901.11173*, 2019.

[58] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, P. S. Yu, Y. Rong, *et al.*, "Fedgraphnn: A federated learning system and benchmark for graph neural networks," *arXiv preprint arXiv:2104.07145*, 2021.

[59] Z. Cai, X. Zheng, and J. Wang, "Efficient data trading for stable and privacy preserving histograms in internet of things," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–10, IEEE, 2021.

[60] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: A self-organized federated learning framework for iot," *IEEE Internet of Things Journal*, 2021.

[61] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "Spreadgnn: Serverless multi-task federated learning for graph neural networks," *arXiv preprint arXiv:2106.02743*, 2021.

[62] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2130–2145, 2021.

[63] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "Fedgnn: Federated graph neural network for privacy-preserving recommendation," *arXiv preprint arXiv:2102.04925*, 2021.

[64] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[65] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang, "Meta-baseline: Exploring simple meta-learning for few-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9062–9071, 2021.

[66] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[67] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[68] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[69] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*, pp. 1126–1135, PMLR, 2017.

[70] N. Ma, J. Bu, J. Yang, Z. Zhang, C. Yao, Z. Yu, S. Zhou, and X. Yan, "Adaptive-step graph meta-learner for few-shot graph classification," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1055–1064, 2020.

[71] J. Chauhan, D. Nathani, and M. Kaul, "Few-shot learning on graphs via super-classes

based on graph spectral measures," in *International Conference on Learning Representations*, 2019.

[72] S. Jiang, F. Feng, W. Chen, X. Li, and X. He, "Structure-enhanced meta-learning for few-shot graph classification," *AI Open*, vol. 2, pp. 160–167, 2021.

[73] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *International Conference on Learning Representations*, 2019.

[74] K. Hassani, "Cross-domain few-shot graph classification," *arXiv preprint arXiv:2201.08265*, 2022.

[75] Y. Lin, X. Wang, H. Ma, L. Wang, F. Hao, and Z. Cai, "An efficient approach to sharing edge knowledge in 5g-enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 930–939, 2022.

[76] Z. Cai and T. Shi, "Distributed query processing in the edge-assisted iot data monitoring system," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12679–12693, 2020.

[77] H. Xu, Z. Cai, R. Li, and W. Li, "Efficient citycam-to-edge cooperative learning for vehicle counting in its," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16600–16611, 2022.

[78] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38-5, pp. 968–979, 2020.

[79] D. F. Nettleton, "Data mining of social networks represented as graphs," *Computer*

Science Review, vol. 7, pp. 1–34, 2013.

[80] X. Zheng, G. Luo, and Z. Cai, "A fair mechanism for private data publication in online social networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 880–891, 2020.

[81] M. Siddula, Y. Li, X. Cheng, Z. Tian, and Z. Cai, "Anonymization in online social networks based on enhanced equi-cardinal clustering," *IEEE Transactions on computational social system*, vol. 6, no. 4, pp. 809–820, 2019.

[82] X. Zheng, Z. Cai, G. Luo, L. Tian, and X. Bai, "Privacy-preserved community discovery in online social networks," *Future Generation Computer Systems*, vol. 93, pp. 1002–1009, 2019.

[83] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow but no track: Privacy preserved profile publishing in cyber-physical social systems.," *Future Generation Computer Systems*, vol. 4, no. 6, pp. 1868–1878, 2017.

[84] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[85] Z. He, Z. Cai, and J. Yu, "Latent-data privacy preserving with customized data utility for social network data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 665–673, 2017.

[86] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2018.

[87] M. Ellers, M. Cochez, T. Schumacher, M. Strohmaier, and F. Lemmerich, "Privacy attacks on network embeddings," *arXiv preprint arXiv:1912.10979*, 2019.

[88] P. Drake, S. Holmes, and S. Redmond, "Who owns celebrity? privacy, publicity, and the legal regulation of celebrity images," *Stardom and Celebrity: A Reader*, pp. 219–29, 2007.

[89] Z. Xiong, Z. Cai, C. Hu, D. Takabi, and W. Li, "Towards neural network-based communication system: Attack and defense," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[90] P. Cockerton, "David beckham and prince william pictured together for campaign against ivory and rhino horn," *https://www.mirror.co.uk/*, 2013.

[91] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[92] K. Li, G. Luo, Y. Ye, W. Li, S. Ji, and Z. Cai, "Adversarial privacy-preserving graph embedding against inference attack," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6904–6915, 2020.

[93] X. Zheng, L. Tian, and Z. Cai, "A fair and rational data sharing strategy toward two-stage industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 1088–1096, 2022.

[94] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[95] Z. He, L. Wang, and Z. Cai, "Clustered federated learning with adaptive local differential privacy on heterogeneous iot data," *IEEE Internet of Things Journal*, 2023.

[96] Z. Xiong, W. Li, and Z. Cai, "Federated generative model on multi-source heterogeneous data in iot," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37-9, pp. 10537–10545, 2023.

[97] C. Wang, Z. Cai, D. Seo, and Y. Li, "Tmeta: Trust management for the cold start of iot services with digital-twin-aided blockchain," *IEEE Internet of Things Journal*, 2023.

[98] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[99] W. Zhang, J. C. Weiss, S. Zhou, and T. Walsh, "Fairness amidst non-iid graph data: A literature review," *arXiv preprint arXiv:2202.07170*, 2022.

[100] K. Zhou, Y. Dong, W. Lee, B. Hooi, H. Xu, and J. Feng, "Effective training strategies for deep graph neural networks. arxiv 2020," *arXiv preprint arXiv:2006.07107*, 2020.

[101] Y. Ye and S. Ji, "Sparse graph attention networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[102] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[103] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[104] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[105] C. Wang and S. Mahadevan, "Manifold alignment using procrustes analysis," in *Proceedings of the 25th international conference on Machine learning*, pp. 1120–1127, 2008.

[106] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[107] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[108] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan, "Cascade residual learning: A two-stage convolutional neural network for stereo matching," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 887–895, 2017.

[109] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36-8, pp. 8485–8493, 2022.

[110] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[111] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

[112] Z. Cai and Z. He, "Trading private range counting over big iot data," in *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, pp. 144–153, IEEE, 2019.

[113] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[114] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[115] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *International Conference on Learning Representations*, 2019.

[116] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec, "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time," in *Proceedings of the 2018 world wide web conference*, pp. 1775–1784, 2018.

[117] S. Kolouri, N. Naderializadeh, G. K. Rohde, and H. Hoffmann, "Wasserstein embedding for graph learning," in *International Conference on Learning Representations*,

2020.

[118] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

[119] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016.

[120] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794, 2015.

[121] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, 2008.

[122] N. Wang, M. Luo, K. Ding, L. Zhang, J. Li, and Q. Zheng, "Graph few-shot learning with attribute matching," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1545–1554, 2020.

[123] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 287–297, Springer, 2008.

[124] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels.," *Journal of Machine Learning Research*, vol. 12,

no. 9, 2011.

[125] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[126] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[127] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[128] F. M. Bianchi, D. Grattarola, and C. Alippi, "Mincut pooling in graph neural networks," 2020.