12-11-2023

Exploring the utility-privacy trade-off in social media data mining

Guangxi Lu

Exploring the utility-privacy trade-off in social media data mining

by

Guangxi Lu

Under the Direction of Zhipeng Cai, Ph.D.

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

Doctor of Philosophy of Science

in the College of Arts and Sciences

Georgia State University

2023

ABSTRACT

Social media data has become an invaluable source of information for data mining. However, developing a high-utility social media model requires a significant amount of training data, which can pose significant privacy challenges. The collection and use of social media data can lead to privacy rights violations and the misuse of personal information, making the trade-off between utility and privacy a complex issue. This dissertation examines the trade-off between utility and privacy in social media data mining from several perspectives. Firstly, it explores how to balance the robustness and fidelity of the social media data mining model in the design of the model structure. Specifically, the study analyzes the use of a pairwise graph convolutional network structure to enhance the model's resistance to adversarial attacks while maintaining accuracy. Secondly, the study examines the trade-off between privacy and utility of social media data in the training framework. To do this, it uses a federated learning framework to investigate the impact of centralizing or decentralizing training on privacy protection and model performance. Finally, the dissertation focuses solely on graph de-anonymization and presents a neural network-based approach to this issue. It explores ways to improve the efficiency and performance of graph de-anonymization through graph embedding vectors. The dissertation also includes a significant number of experiments to validate the feasibility of the proposed framework from both utility and privacy perspectives. The results demonstrate that an appropriate model or framework design can reasonably balance the privacy and utility of social media data mining.

INDEX WORDS: Social Media, Utility-Privacy Trade-off, Adversarial Attacks, Federated Learning

Exploring the utility-privacy trade-off in social media data mining

by

Guangxi Lu

Committee Chair: Zhipeng Cai

Committee: Yingshu Li

Wei Li

Yan Huang

Electronic Version Approved:

Office of Graduate Services

College of Arts and Sciences

Georgia State University

December 2023

DEDICATION

This dissertation is dedicated to my parents, Ming Guo and Jianchuan Lu, who have always supported and encouraged me throughout my academic journey, thanks for their love, guidance and unwavering support. Their sacrifices and hard work have made it possible for me to pursue my dreams. Thanks for my advisor, Zhipeng Cai, for his invaluable guidance, support and mentorship throughout my research. I am grateful for the opportunity to work under his supervision and for the wealth of knowledge he has shared with me. Thanks for my colleagues and friends, for their camaraderie and support throughout the years. And lastly, Thanks for all those who have helped me in one way or another, I am deeply grateful. This thesis is dedicated to all of you.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to the following individuals and organizations who have contributed to the completion of this thesis. My advisor, Dr. Zhipeng Cai , for his unwavering support, guidance and mentorship throughout my research journey. I am grateful for the opportunity to work under his supervision and for the wealth of knowledge he has imparted to me. Dr. Wei Li. I am deeply grateful to her. Without her expert supervision and support, I would not have been able to complete this thesis. Her patience and guidance were invaluable and have played a critical role in my academic journey. My colleagues, friends and fellow graduate students, Zuobin Xiong, Honghui Xu, Kainan Zhang, Peng Wang, Kaiyang Li, and Yuchen Wang for their camaraderie, support and helped me a lot in study and live. The computer science department at georgia state university, for providing the resources, support and facilities that were essential to my research. Thanks to Dr. Xu Zheng and Dr. Ling Tian for their assistance and support. And lastly, to all those who have helped me in one way or another, I am deeply grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

With the rapid development of the internet, social media platforms such as Facebook, Twitter, and Instagram have changed the way people communicate [107]. These platforms generate a huge amount of data every day. Through data mining technology, these information can be effectively extracted and used for user behavior [27], emotion [102], and public opinion analysis [87]. As a result, many organizations have recognized the importance of data mining,[93] as it not only helps them better understand user needs, but also enables them to make more informed decisions and improve operational efficiency.

However, the growing use of social media data mining has raised privacy concerns [37][90]. The massive amount of personal information collected by social media platforms makes it vulnerable to exploitation by third-party organizations. This raises the alarm for users about the privacy of their personal information and the potential abuse of their data. [11] Unfortunately, ensuring user privacy and the utility of social media data mining is a trade-off. [14] The need to protect personal information often conflicts with the need to extract valuable insights from the data, making it difficult to strike a balance between privacy and utility [98][10].

Balancing privacy and utility in social media data mining presents two key challenges. [21][17] Firstly, models used to mine the graph structure of social media data often lack robustness, making them susceptible to attacks by malicious user. On the other hand, designing a high robustness model can introduce noise or redundant structures, undermining the accuracy

and fidelity of the model. Secondly, the selection of a training framework also presents difficulties. Centralized training increases the risk of data being leaked to third parties, while decentralized training requires a significant amount of communication and may result in a decline in model accuracy.

This dissertation aims to analyze the trade-off between model utility and privacy from both the data mining model and training framework perspectives. Specifically, at the model level, the dissertation proposes a pairwise graph convolutional network that maintains robustness while ensuring model fidelity. [60] At the training framework level, the paper introduces a decentralized federated learning approach and compares its privacy protection against gradient attacks with other centralized training methods. [59][58]

To further explore social media data privacy, this study delves into data-level privacy protection by proposing a neural network-based graph de-anonymization algorithm. The aim is to achieve a more efficient and accurate implementation of social data de-anonymization, providing valuable insights into the privacy protection of social media data and informing future research in this field.

The remaining parts of this dissertation are organized as follows. Chapter 2 summarizes relevant literature. Chapter 3 investigates the trade-off between privacy protection and utility at the model level. Chapter 4 examines the trade-off between privacy protection and utility at the training framework level. Chapter 5 introduces the data level graph de-anonymization research. Chapter 6 introduce the future work. Chapter 7 is the conclusions of this dissertation.

CHAPTER 2
BACKGROUND

Social media data has become an indispensable resource in today's society. It captures the information people post on social media platforms, including text, images, audio, and video[57], providing valuable insights into people's lifestyles, interests, values, and information consumption habits. Utilizing this data allows us to better understand people's behavior and psychology, population characteristics and consumption trends, as well as to express ourselves and communicate with others.

One of the main aspect of social media data is that it is presented in the form of a graph. A graph is a mathematical representation of a set of objects and their relationships. In social media, objects can be users, and relationships can be the connections between users, such as friendships or followings. The graph structure of social media data allows for capturing complex relationships between users. Therefore, graph convolutional networks are commonly used for analyzing these social media data. Another aspect of social media data is that it is large in scale and widely dispersed with heterogeneous sources[65]. To achieve joint training across different data sources while preserving privacy, federated learning frameworks are often used. Therefore, this dissertation focuses on exploring the trade-off between performance and privacy in graph convolutional networks and federated learning frameworks.

2.1 Graph Convolutional Networks

Graph Convolutional Networks(GCN) is a graph embedding technique that maps graph structures into vectors by combining convolutional neural networks and exploiting the struc-

tural information of the graph. The main idea of GCN is to extract the spatial features of the graph structure data. Graph convolutional networks follow a "message-passing" frameworks. Specifically, each node in the network aggregates information from its neighbors nodes through convolution layer. The information in the graph structure is continuously aggregated until all the information converges. Essentially, GCN is the study of the properties of a graph with the help of the eigenvalues and eigenvectors of the laplace matrix. Bruna et al.[5] first introduce the idea of use convolutional in graph data. It generalized CNN to graph structure data based on the spectrum of the graph laplacian. Defferrard et al[22] present a formulation of CNNs in the context of spectral graph theory and design a fast localized convolutional filters on graphs. Kipf and Welling[45] propose graph convolutional networks. It is a graph neural networks used for semi-supervised classification in graph structured data, and it is actually a first-order approximation of spectral graph convolution.

In addition, GCN also has several vertex variants. Li et al.[53] use gated recurrent units extend to output sequence, and then aggregate graph structure information. Peter et al.[77] introduces an attention mechanism to GCN. By assigning different learning weights to different node, makes it more reasonable in dealing with the directional graph tasks. However, all of these works do not consider the robustness of the GCN models. so they are vulnerable to adversarial attacks.

Anyway, GCN stands out in the realm of graph mining. It effectively compresses the rich, high-dimensional neighborhood data of nodes into concise vector representations, enabling deeper pattern recognition than traditional techniques. This capability, rooted in

deep learning, permits GCNs to discern intricate relationships and dynamics within graphs.

However, GCNs are not without challenges. Their impressive performance is offset by a marked vulnerability to adversarial attacks [20]. Adversaries can subtly manipulate graph data, such as tweaking edges or node attributes, compromising the integrity of GCN outputs. These perturbations can lead to significant misclassifications, posing potential security and privacy risks.

The inherent messaging-passing framework of GCN amplifies this concern. A slight alteration in one node can propagate, influencing neighboring nodes and, by extension, the entire graph. This interdependency complicates the defense against adversarial attacks.

Current defensive strategies are still maturing. One approach involves detecting and rectifying perturbations, but given the interconnected nature of GCNs, identifying these with precision is challenging. Furthermore, removing all perturbed features might impair the model's efficacy. Another proposition is to encapsulate node feature values within a Gaussian distribution. While this might dampen the impact of perturbations, it introduces noise, potentially affecting the model's accuracy on clean datasets.

In essence, the journey to harmonize the robustness and performance of GCNs remains a topical research challenge.

2.2 Federated Learning

In the evolving landscape of machine learning, the role of large datasets for effective model training is pivotal. However, many organizations grapple with the issue of data residing in

isolated silos, a consequence of pressing privacy concerns, competitive barriers, and other regulatory constraints.

Addressing this challenge, Federated Learning (FL) has emerged as a potent framework. FL offers a collaborative mechanism for multiple entities to jointly train models without compromising the sanctity of individual data points. By circumventing direct data sharing, FL presents a solution to the data silo problem, and its applications span sectors like electronic finance, recommendation systems, and biomedicine.

Consider a scenario involving N distinct entities or enterprises, each possessing its own localized data. Direct data interchange among them is hindered by privacy and security protocols. Here, the FL system fills the gap. Each entity, termed a client, leverages its data to train localized models, subsequently transmitting the trained parameters to a central server. This server amalgamates the inputs from all clients, updating a global model. The process is iterated until satisfactory training of the global model is achieved.

Drawing a parallel, when companies aim to develop a collective machine learning model, each initially focuses on localized model training. Instead of dispatching raw data, periodic interactions with a central server occur. In these interactions, local model parameters are shared, with the server reciprocating by dispatching an updated global model. Iterations ensue until a convergence criterion is met. Crucially, FL ensures that the local data remains confined, obviating direct data transfers and thereby ensuring robust data security.

Google first introduced the concept of federated learning in 2016 as a means of updating machine learning models locally for Android users. The federated averaging proposed by

[63] follows a synchronous training approach in which each client trains a local model and sends it to a centralized server, which computes the average of all models. The server sends the average model back to the clients. Numerous approaches have been proposed to enhance privacy and security in a federated learning environment[86] applied differential privacy to provide the necessary privacy guarantees whereas proposed a secure aggregation protocol for privacy-preserving machine learning in a federated environment.[4] built a production system, the first of its kind, for federated learning in the field of mobile devices using TensorFlow.

Nonetheless, FL is not without its vulnerabilities. Despite its design, which promotes model training sans direct data sharing, recent inquiries highlight potential risks. These risks manifest when servers, possessing model gradients, deploy targeted attacks to infer aspects of the client's private data. For instance, through membership inference attacks, it's possible to identify specific data points involved in the local model's training. Similarly, label inference attacks and model inversion attacks can deduce particular labels and regenerate supposedly private local data samples, respectively.

A proactive approach to mitigate such risks involves the elimination of the central server from the FL framework. A nascent concept, Decentralized Federated Learning (DFL), champions this shift. In DFL, there's an absence of a central data aggregation point. Instead, a peer-to-peer (P2P) structure is adopted, wherein clients directly liaise with neighboring peers. Here, individual clients train their models, relay parameters across the P2P network, and update their models based on received feedback. This decentralized approach curtails the consistent extraction of gradient information by potentially malicious clients, fortifying

data privacy. Concurrently, to maintain the efficacy of DFL, novel training strategies have been proposed to harmonize model accuracy with convergence efficiency.

In summation, while Federated Learning represents a significant advance towards reconciling data utility with privacy, the pathway presents both challenges and prospects. This underscores the imperative for ongoing research and innovation in decentralized learning paradigms.

2.3 Graph De-anonymisation

To further explore the privacy protection of social network data, a pivotal idea is delving into the privacy of graph data itself. Graph data structures have become the backbone of a multitude of applications, ranging from social networks to biological systems. As these applications often involve sensitive information, ensuring the privacy of individuals and entities represented in these graphs has become a critical concern. To protect the identities and relationships of individuals in a graph, various anonymisation techniques have been developed. These techniques aim to obscure specific node and edge details while retaining the overall graph's structural information. Methods like k-anonymity [72] ensure that each node in the graph cannot be distinguished from at least k-1 other nodes based on their attributes. Similarly, techniques like l-diversity [62] aim to ensure that sensitive label information is obscured among l distinct entities. Despite the advancement of these anonymisation methods, several graph de-anonymisation techniques have emerged. These techniques attempt to unveil the original identities of nodes in a graph by leveraging structural information and

external data. For instance, if an adversary possesses a fraction of the original graph or some auxiliary information, they might be able to align the anonymised graph's nodes to the original nodes based on specific patterns or similarities [42]. The effectiveness of graph de-anonymisation attacks highlights the vulnerabilities inherent in relying solely on graph modification as a means of preserving privacy. Successful de-anonymisation can lead to significant privacy breaches, where sensitive information about individuals or organizations is exposed. Such revelations have implications not just for personal privacy but also for business confidentiality, national security, and other domains. The constant tug-of-war between anonymisation and de-anonymisation has led to a dynamic landscape. Researchers are continuously developing more robust anonymisation techniques while also uncovering new vulnerabilities. This has led to a growing awareness of the need for a multi-faceted approach to graph data privacy, which combines structural modifications, noise addition, and possibly differential privacy guarantees.

CHAPTER 3
DATA MINING MODEL

3.1 Introduction

With the help of graph representation, the topology of wireless networks and the attributes of nodes (*e.g.*, locations, mobile users, and sensors) can be comprehensively and accurately depicted for data analysis using graph mining, such as node classification, clustering, link prediction, and recommendation. These mining results play a critical role in many real-world applications, including transmission schedule in wireless networks [104][6], offloading decisions in edge computing [75][9], behavior forecasting from sensor data [15][97], traffic monitoring [16][89], epidemic forecast [81], *etc.*

As a promising technique of graph mining, GCN can capture the graph global information by downscaling nodes' high-dimensional neighborhood information to low-dimensional vector representations, where notably, deep learning mechanisms can learn and discover more complex patterns in graphs than traditional graph mining algorithms.

Despite the remarkable performance of GCN, it is vulnerable to adversarial attack [20][94][52] due to the lack of robustness [19]. Through imperceptible modifications on graph data via adding edges, deleting edges, or changing node attributes, attackers intend to mislead classification/prediction results so as to reduce the performance of GCN. What's worse, such mis-classified results will cause serious security and privacy threats towards downstream applications.

Even though the consequence of adversarial attack on GCN has caught the attention

of researchers, the study on defense strategies is still at its early stage. The difficulty of adversarial defense on GCN is that GCN is a messaging-passing framework [110]. Therefore, a perturbation of one node affects his neighboring nodes, thus affecting all nodes in the graph during the messaging-passing process.

Among the current works, one pioneer approach to resist adversarial attacks on GCN is to detect and eliminate perturbations [25]. However, due to the messaging-passing property of GCN, a slight perturbation can alter many node features in a graph. Therefore, it is challenging to detect perturbations on a graph accurately. On the other hand, eliminating all the altered features would significantly reduce the model's classification accuracy. Another approach is to represent the feature values of nodes by gaussian distribution [110], so that the effects from perturbations are passively absorbed in this gaussian structure. But, due to the additional noise introduced by Gaussian distribution, the Gaussian framework inevitably leads to a decrease in the classification accuracy of GCN models when dealing with unperturbed datasets. Therefore, how to minimize the impact of perturbations under adversarial attacks while ensuring the classification accuracy of GCN models remains an open problem.

In order to improve model robustness while simultaneously ensuring model accuracy, this paper develops a novel GCN-based framework named Pairwise Gaussian Graph Convolutional Network (PGGCN). Such a pairwise structure comprises a Gaussian GCN and a Perturbed GCN. The Gaussian GCN takes Gaussian distribution as the hidden representations of node features/attributes to sample the original node features, and the Perturbed GCN employs the perturbed data as the input to obtain the perturbed node features. Based

on these outputs, the loss of difference between the original node features and the perturbed node features is formulated to improve the model robustness, and the loss of classification on the original node features is formulated to enhance the model accuracy.

For performance evaluation, we conduct extensive real-data experiments and find that our PGGCN outperforms state-of-the-art GCN under various adversarial attack strategies. The main contributions of this paper are summarized as follows.

- To the best of our knowledge, we are the first to propose the idea of pairwise structure to design GCN models for the defense against graph adversarial attack.
- In our proposed PGGCN framework, graph features are shared between the Gaussian GCN and the Perturbed GCN to estimate classification loss and differential loss, which is used in the training process for enhancing model robustness and ensuring classification accuracy.
- Various real datasets, state-of-the-art models, and graph adversarial attacks are utilized to carry out experiments, which validates the advantages of our PGGCN framework in model robustness and classification accuracy.

The rest of this paper is organized as follows. The existing works in related fields are summarized in Section 3.2. The preliminaries are introduced in Section 3.3. Section 3.4 presents our PGGCN model. Experiments are conducted and analyzed in Section 3.5. Finally, this paper is concluded in Section 3.6.

3.2 Related Work

This section reviews current trends and main achievements on adversarial attacks and defense in GCN.

3.2.1 Graph Adversarial Attack

Graph adversarial attack is the process of applying a slight perturbation to the graph data to generate adversarial samples. Such adversarial samples can mislead the victim model like Graph Convolutional Networks(GCN) to produce incorrect classification results. Graph adversarial attack can add perturbations in various ways, such as modifying node features, adding/deleting edges, and adding fake nodes. The perturbation should be less than a certain threshold when attacking. Otherwise, it would be easily detected.

Gradients are commonly exploited to attack victim models, where the most commonly used gradient methods are the Fast Gradient Notation Method (FGSM)[28] and the Jacobian-based Salinity Map Method (JSMA)[67]. FGSM generates adversarial examples by performing gradient updates along with the sign of gradients of the loss function. JSMA exploiting the forward derivative of GCN, one can find the adversarial perturbations that force the model to misclassify the test point into a specific target class. In addition, many of the adversarial attack methods used in the image domain can also be applied to graph data.[96] Dai et al.[19] propose a reinforcement learning-based graph adversarial attack method, which treats the adversarial attack as a finite layer Markov decision process and uses Q-learning to learn this process. Daniel et al.[114] propose an optimization-based algorithm named Net-

tack. This algorithm takes into account the dependencies between instances and generates adversarial perturbations by searching the perturbation space. Daniel et al.[115] also propose an attack method named Meta-attack, which first gets the target label from the victim model by self-learning and then makes the prediction of the post-attack model as different as possible from that target label. Yao et al.[61] propose the ReWatt framework based on the idea of RL-S2V and rewiring. Rewiring does not change the number of nodes and edges in the graph, further reducing perturbations.

Graph adversarial attacks have posed great challenges to the robustness of GCN, which severely limits the applicability of GCN in real world applications. How to ensure both GCN accuracy and robustness is the focus of this paper.

3.2.2 Graph Adversarial Defense

Currently, GCN-based adversarial defense methods are still in their early stage. Wang *et al.* [83] proposed an adversarial training algorithm called GraphDefense that injects adversarial samples into the training dataset so as to help victim models correctly classify future adversarial samples, in which iterative training is required, and there is only a slight improvement in robustness. Xu *et al.* [100] stated that adversarial attack can be resisted by detecting adversarial perturbations. They predicted the probability of two nodes being linked by the link prediction. If this probability is low, it means attackers add the corresponding edge. This approach can filter out the perturbed data but does not improve model robustness. Shen *et al.* [82] investigated the latent vulnerabilities in every layer of GCN. The GCN robustness is improved by using a two-stage aggregation structure and adding a bottleneck

aggregator. However, the change in structure also leads to a decrease in the classification accuracy of GCN. Zhu *et al.* [110] proposed Robust Graph Convolutional Networks by introducing an attention mechanism with the Gaussian based structure. RGCN are trained by penalizing the weights on adversarial edges or nodes, where the introduced additional noise can enhance the GCN robustness while reducing prediction accuracy.

Wei *et al.* [43] found that adversarial attack usually increases the singular value and rank of the adjacency matrix or tends to connect two nodes with different features. Therefore, a Pro-GNN network is proposed to reconstruct the Poisoned Graph to maintain the graph data with low rank, sparsity, and smoothness of features. Entezari *et al.* [23] pointed out that Nettack [114], a state-of-the-art adversarial attack method, only affects high-rank singular components of a graph. Based on this observation, the low-rank approximation can be used to filter the noise brought by Nettack. However, there is no reliable evidence that the high-rank phenomenon is prevalent in all graph adversarial attacks.

Nevertheless, few of the current graph adversarial defense methods can improve the robustness of GCNs with the guarantee of model accuracy.

3.3 Preliminaries

Graph Convolutional Network (GCN) is a well-designed method for extracting features from graph data to perform node classification [35]. A graph can be represented as $G = (V, E)$, where V is the set of nodes, and E is the set of edges. For all the nodes in G, $X \in \mathbb{R}^{m \times n}$ is the node feature matrix, $D \in \mathbb{R}^{n \times n}$ is the node degree matrix, and $A \in \mathbb{R}^{n \times n}$ is a adjacency

matrix, where n denotes the number of nodes, and m denotes the dimensionality of the node feature.

The simplest form of GCN can be described as

$$H^{l+1} = \sigma\left(AH^lW^l\right), \tag{3.1}$$

where the $H^l \in \mathbb{R}^{m\times n}$ represents the feature map of layer l, and $H^0 = X$. W^l is the trainable parameters. σ is any nonlinear activation function, such as sigmoid or RELU. When the feature map H^l multiplies the adjacency matrix A, the feature vector of each node is added to their adjacent vectors. This operation causes the feature of each node to pass to all one-hop neighborhoods and thus deliver the message. One limitation of this model is the lack of self-deliver of node information at each GCN iteration because the diagonal values of the adjacency matrix are all 0. This problem can be solved by adding the identity matrix to A, which is $\hat{A} = I + A$.

According to Kipf [45], GCN can be normalized to the most common form:

$$H^{l+1} = \sigma\left(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}H^lW^l\right), \tag{3.2}$$

where $H^l \in \mathbb{R}^{m\times n}$ represents the feature map at layer l (especially, $H^0 = X$), W^l is the trainable parameters, σ is any nonlinear activation function (such as sigmoid and RELU), and $\hat{A} = I + A$.

After the input data has gone through several GCN layers, the graph achieves adequate node information transfer. The predicted classification probability can be directly calculated

as

$$Z = f_\theta(A, X) = \text{softmax}(H^t), \tag{3.3}$$

where $f_\theta(A, X)$ represents a GCN model,θ denotes the set of all parameters in GCN. H^t is the node feature in the last GCN layer t. This feature passed through a softmax function to finally obtain the predicted classification results Z. The cross-entropy loss is taken to calculate the difference between the predicted classification result and the true value, such as:

$$L(\theta; A, X) = -\sum_{v \in V_l} \sum_{c=1}^{C} Y_{vc} \ln Z_{vc}, \tag{3.4}$$

where V_l is the set of labeled node, C is the number of classes, Y_{vc} is the ground truth of labeled nodes v, and Z_{vc} denotes the probability of node v to label c.

Adversarial attack on GCN aims to cause an incorrect prediction for node classification by adding perturbation, such as adding/deleting edges and changing node features. Among the existing works, Nettack [114] is the state-of-the-art adversarial attack method. The attack of Nettack is shown in Fig. 3.1. The attacker causes the classification result of target node V_4 change from Y_4 to Y_4' by adding the edges $< V_2, V_3 >$. Nettack assume the original graph is represented by a binary group (A, X), which X is the feature map and A is the adjacency matrix. The modified graph is $G' = (A', X')$. The nodes modified by the attacker form the attacking nodes set V_a which $V_m = \{V_2, V_3\}$ in Fig. 3.1. The perturbations are hold that[114]:

$$X'_{ui} \neq X_{ui} \Rightarrow u \in V_m \quad , \quad A'_{uv} \neq A_{uv} \Rightarrow u \in V_m \vee v \in V_m, \tag{3.5}$$

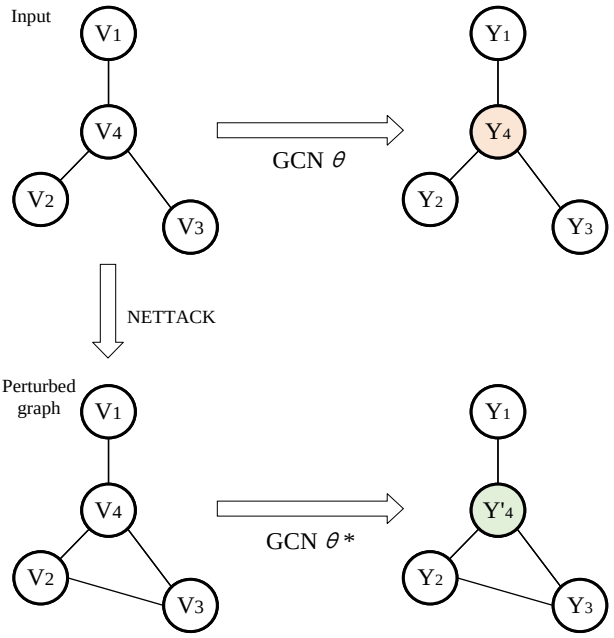


Figure 3.1: Adversarial attack in graph structure data

where i is the element of feature vector. The prerequisite for the attack is that the perturbation is undetected, So there is a budget constraint Δ on the attack model, which is[114]:

$$\sum_{u=1}^{n}\sum_{i=1}^{m}|X_{ui}-X'_{ui}|+\sum_{u=1}^{n}\sum_{v=1}^{u}|A_{uv}-A'_{uv}|\leq\Delta. \tag{3.6}$$

For the input graph G, there exists a perturbation space P^{G}_{Δ,V_m} containing all possible perturbations. For any target node is v, the original GCN classifies v with the label c_{old}, while the modified GCN classifies v as the label c_{new}. The idea of Nettack is to maximize the GCN classification loss (*i.e.*, the classification distance between the original and the modified GCN models) on the target node through perturbations in P^{G}_{Δ,V_m}, which can be expressed

below [114]:

$$\begin{gathered}\underset{(A',X')\in P^{G}_{\Delta,V_m}}{\arg\max}\ \max_{c_{\text{new}}\neq c_{\text{old}}}\ \ln Z^{*}_{vc_{\text{new}}}\ -\ln Z^{*}_{vc_{\text{old}}}\,, \\ \text{s.t., } Z^{*}=f_{\theta^{*}}\left(A',X'\right)\ \text{ with } \theta^{*}=\arg\min_{\theta} L\left(\theta;A',X'\right),\end{gathered} \tag{3.7}$$

where θ^{*} is the optimal parameters trained from the modified graph $G'=(A',X')$. To measure the distance between these two labels, Nettack has built a simple proxy GCN model. Specifically, Nettack builds a two-layer GCN that:

$$Z=f_{\theta}(A,X)=\text{softmax}\left(\hat{A}\sigma\left(\hat{A}XW^{1}\right)W^{2}\right). \tag{3.8}$$

By removing the nonlinearity σ and the softmax function, the classification result of this two-layer GCN can be simplified to:

$$lnZ=[\hat{A}^{2}XW]. \tag{3.9}$$

Thus, Nettack can score the perturbations present in perturbation space P^{G}_{Δ,V_m}, and this scoring function is the maximum classification distance generated by that perturbation as[114]:

$$S=L\left(A,X;W,v\right)=\max_{c_{\text{new}}\neq c_{\text{old}}}\left[\hat{A}^{2}XW\right]_{vc_{\text{new}}}-\left[\hat{A}^{2}XW\right]_{vc_{\text{old}}}. \tag{3.10}$$

Nettack uses a greedy algorithm to select the perturbations that need to be added. The distance caused by each perturbation is calculated through Eq. (3.10). Each time, the perturbation with the largest distance is selected for addition and the process is repeated until the budget is exceeded.

3.4 Methodologys

In order to enhance the robustness of GCNs against adversarial attacks, we propose a novel model, termed Pairwise Gaussian Graph Convolutional Network (PGGCN) as shown in Fig. 3.2. Our proposed PGGCN is a pairwise structure consisting of a Gaussian GCN and

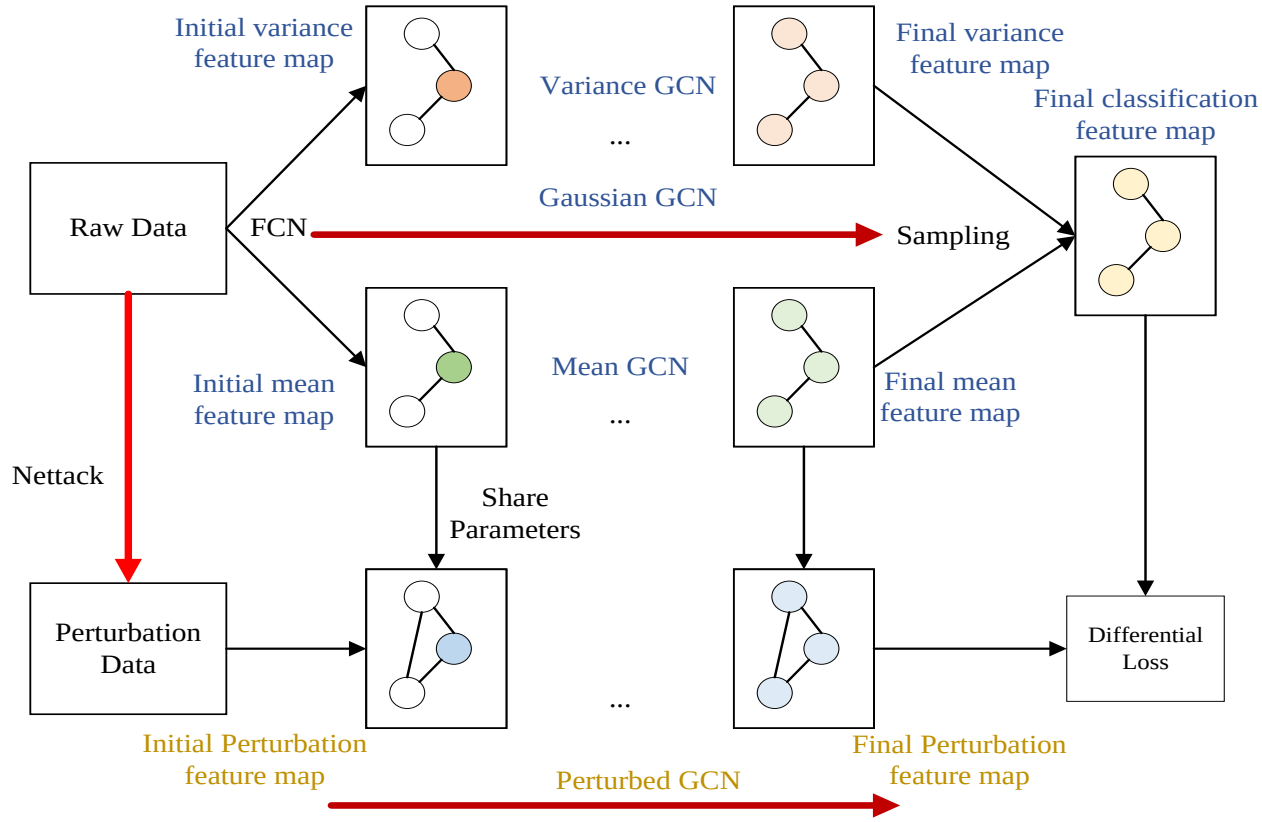


Figure 3.2: The framework of Pairwise Gaussian Graph Convolutional Network

a Perturbed GCN. The Gaussian GCN, which contains a mean GCN and a variance GCN, first maps the feature of raw data into the initial mean feature map and the initial variance feature map through two fully connected layers (*i.e.*, FCN). These two feature maps are passed through the mean GCN and the variance GCN to obtain the final mean feature map

and final variance feature map, respectively, and then sampled to get the final classification feature map. The Perturbed GCN shares the model parameters with the mean GCN, In the Perturbed GCN, the input is the perturbed data obtained by performing Nettack on the raw data, and the output the final perturbed feature map. This pairwise structure enhances model robustness by minimizing the difference between the perturbed features and the classification features. In the following, we will introduce how to realize our models in detail.

3.4.1 Gaussian-based Graph Convolution Network

The core idea of the Gaussian GCN is to represent the feature values by gaussian distribution; that is, the raw data feature is represented by the mean feature map and the variance feature map. The raw data feature map is denoted as $H^0 = [h_1^0, \ldots, h_n^0] \in \mathbb{R}^{m \times n}$, where h_i^0 is an m-dimension feature vector of node i. Assume that the feature vectors of the raw data follow gaussian distribution, *i.e.*, $h_i^0 \sim N\left(\mu_i^0, \operatorname{diag}\left(\tau_i^0\right)\right)$, where $\mu_i^0 \in \mathbb{R}^m$ is the mean vector, and $\tau_i^0 \in \mathbb{R}^m$ is the variance vector. Accordingly, $E_i^0 = [\mu_1^0, \ldots, \mu_n^0] \in \mathbb{R}^{m \times n}$ and $S_i^0 = [\tau_1^0, \ldots, \tau_n^0] \in \mathbb{R}^{m \times n}$ are the means and variances feature maps. Therefore, the first layer of Gaussian GCN can convert the raw data feature maps into mean and variance feature maps via two fully connected layers (FCN).

$$E^0 = \sigma\left(H^0 W_\mu^0\right), \tag{3.11}$$

$$S^0 = \sigma\left(H^0 W_\tau^0\right), \tag{3.12}$$

where W_μ^0 is the trainable parameters of the mean FCN, and W_τ^0 is the trainable parameters of the variance FCN.

The Gaussian GCN aggregates mean and variance feature based on Eq. (3.2) by applying learnable filters and non-linear activation functions, which directly imposes layer-specific parameters and non-linear activation functions to mean feature E and variance feature S. Formally, in PGGCN, Gaussian-based graph convolution is defended as follows:

$$E^{l+1} = \sigma\left(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}E^l W_\mu^l\right), \tag{3.13}$$

$$S^{l+1} = \sigma\left(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}S^l W_\tau^l\right), \tag{3.14}$$

where W_μ^l is the trainable parameters of the mean GCN in layer l, and W_τ^l is the trainable parameters of the variance GCN in layer l.

From Eq. (3.2), we have the following expression for GCN:

$$\begin{aligned} h_i^{l+1} &= \sum_{j=1}^{n} D_{ii}^{-1/2}\hat{A}_{ij}D_{jj}^{-1/2}h_j^l W^l, \\ &= \sum_{j=1}^{n} \frac{\hat{A}_{ij}}{D_{ii}^{1/2}}\frac{h_j^l}{D_{jj}^{1/2}}W^l, \end{aligned} \tag{3.15}$$

where j and i are two nodes in the graph G, and n is the number of node. Therefore, the essence of GCN is a weighted sum process. which can be rewritten as:

$$h_i^{l+1} = \sum_{j=1}^{n} w_j h_j^l, \tag{3.16}$$

where $w_j = \frac{\hat{A}_{ij}}{D_{ii}^{1/2}}\frac{W^l}{D_{jj}^{1/2}}$ is the weight.

Theorem 1. *If $\alpha \sim N\left(\mu_\alpha, \tau_\alpha^2\right)$ and $\beta \sim N\left(\mu_\beta, \tau_\beta^2\right)$ are two statistically independent random variables following gaussian distribution, the weighted sum of these two variables also satisfies gaussian distribution:*

$$W_\alpha \alpha + W_\beta \beta \sim N\left(W_\alpha \mu_\alpha + W_\beta \mu_\beta, W_\alpha^2 \tau_\alpha^2 + W_\beta^2 \tau_\beta^2\right). \tag{3.17}$$

Proof. As $\alpha \sim N\left(\mu_\alpha, \tau_\alpha^2\right)$ and $\beta \sim N\left(\mu_\beta, \tau_\beta^2\right)$, the moment-generating functions of α and β should be:

$$M_\alpha(t) = e^{\left(\mu_\alpha t + \frac{\tau_\alpha^2 t^2}{2}\right)}. \tag{3.18}$$

$$M_\beta(t) = e^{\left(\mu_\beta t + \frac{\tau_\beta^2 t^2}{2}\right)}. \tag{3.19}$$

The moment-generating function of $W_\alpha \alpha + W_\beta \beta$ can be computed as follows.

$$\begin{aligned} M_{Sum}(t) &= M_\alpha\left(W_\alpha t\right) * M_\beta\left(W_\beta t\right), \\ &= e^{\mu_\alpha W_\alpha t} * e^{\mu_\beta W_\beta t} * e^{\frac{W_\alpha^2 \tau_\alpha^2 t^2}{2}} * e^{\frac{W_\beta^2 \tau_\beta^2 t^2}{2}}, \\ &= e^{(\mu_\alpha W_\alpha + \mu_\beta W_\beta) t + \frac{(W_\alpha^2 \tau_\alpha^2 + W_\beta^2 \tau_\beta^2) t^2}{2}}. \end{aligned} \tag{3.20}$$

Eq. (3.20) shows that the moment-generating function of $W_\alpha \alpha + W_\beta \beta$ is the same as the moment-generating function of a normal random variable with $W_\alpha \mu_\alpha + W_\beta \mu_\beta$ being the mean and $W_\alpha^2 \tau_\alpha^2 + W_\beta^2 \tau_\beta^2$ being the variance. □

Since the first layer feature h_i^0 follows gaussian distribution, the final layer feature h_i^f should also satisfy gaussian distribution according to Theorem 1. Finally, the final classification feature map can be obtained through the mean feature map and variance feature

map, and the sampling method will be used to get the classification features.

$$h_i^f \sim N\left(\mu_i^f, \mathrm{diag}\left(\tau_i^f\right)\right). \tag{3.21}$$

However, the sampling operation makes μ_i^f and τ_i^f integrable. In order to back-propagate their gradients, Kingma *et al.* [44] proposed reparameterize trick to rewrite h_i^f as $h_i^f = \tau_i^f \epsilon + \mu_i^f$, where $\epsilon \sim N(0,1)$. The random quantities can be sampled from the standard gaussian distribution and then introduced into μ_i^f and τ_i^f.

3.4.2 Pairwise Framework

The main idea of our pairwise framework comes from adversarial training – the robustness of GCN can be improved by minimizing the distance between the original and perturbed graph features. As shown in Fig. 3.3, the PG-GCN forms a conjoined structure with a Gaussian-based GCN as the trunk and shares weight with a branching GCN. The trunk

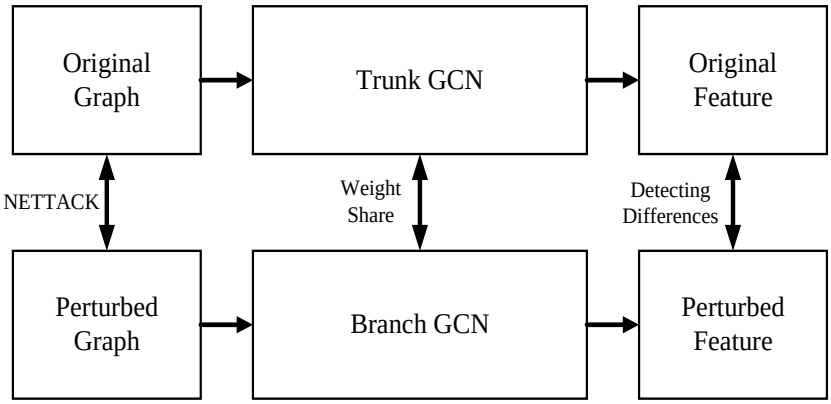


Figure 3.3: The pairwise differential structure

GCN designed to extract the original feature for the original graph data. Meanwhile, a perturbated graph is formed after adversarial attacks on the original data. The branch GCN

extracts the perturbated features from this perturbation graph. For the gaussian-based GCN, the variance component represents the perturbation that the model can tolerate. The larger the variance, the more uncertain the feature values are, and the greater the robustness of the model. The mean component represents the most likely classification result of the model. Therefore, the branching GCN shares the model parameters from the mean component W_μ.

In order to detect the difference of graph features before and after attack, the raw data is first subjected to Nettack to obtain the perturbed data $G' = (V', E')$, where V' is the set of nodes after attack, and E' is the set of edges after attack. To minimize the differences in the feature maps before and after perturbation, the model parameters of the Perturbed GCN should be equal to those of the Gaussian GCN. On the other hand, the Gaussian GCN structure introduces additional randomness by using the variance GCN, which may reduce the model accuracy of the Perturbed GCN. Thus, the Perturbed GCN only shares parameters with the mean GCN and then obtains the final perturbed feature map H'^f as:

$$H'^f = \sigma\left(D'^{-\frac{1}{2}}\hat{A}'D'^{-\frac{1}{2}}H'^{f-1}W_\mu^{f-1}\right), \tag{3.22}$$

where $D' \in \mathbb{R}^{n\times n}$ is the node degree matrix after attack, $A' \in \mathbb{R}^{n\times n}$ is the adjacency matrix after attack, W_μ^{f-1} is the mean GCN parameters in the layer $f-1$. Especially, $H'^0 = X'$, where $X' \in \mathbb{R}^{m\times n}$ is the node feature matrix after attack,

Our PGGCN mechanism aims at minimizing the feature distance to mitigating the effect of perturbation so as to improve model robustness. In this paper, we use L_1 distance to calculate this difference:

$$L_{\text{diff}}(\theta; G', G) = \left\|H^f - H'^f\right\|_1. \tag{3.23}$$

where θ denotes the set of all trainable parameters in PGGCN, *i.e.*, $\theta = (W_\tau, W_\mu)$. Moreover, to ensure the accuracy of the GCN model, PGGCN also needs to measure the classification loss that is defined as:

$$L_{\text{cls}}(\theta; G) = -\sum_{v \in V_l} \sum_{c=1}^{C} Y_{vc} \ln Z_{vc}, \tag{3.24}$$

where V_l is the set of labeled nodes, C is the number of classes, Y_{vc} is the ground truth of labeled nodes v. $Z_{vc} = \text{softmax}(H^f)$ denotes the probability of node v being classified to label c.

Finally, by taking into account both the feature distance and the classification loss, the total loss function of PGGCN can be obtained:

$$L_\theta = \lambda L_{\text{diff}}(\theta; G', G) + (1 - \lambda) L_{\text{cls}}(\theta; G). \tag{3.25}$$

In Eq. (3.25), L_{diff} assesses how far an adversarial feature is away from its original value, which is used to ensure the model robustness; L_{cls} evaluates the importance of each feature element for classification, which helps guarantee the model accuracy; and $\lambda \in [0, 1]$ is a hyper-parameter to balance the trade-off between L_{diff} and L_{cls}.

3.5 Performance Evaluation

This section presents real-data experiments to evaluate the performance of our PGGCN framework.

Table 3.1: Basic information of the datasets

	Cora	Citeseer	Pubmed
# of Nodes	2708	3327	19717
# of Edges	5429	4732	44338
Feature Dimension	1433	3703	500

3.5.1 Datasets

For node classification, three different datasets are adopted, including Cora, Citeseer, and Pubmed as detailed in Table 3.1. Each dataset is divided into a labeled set (containing 95% of the entire dataset) and an unlabeled set (containing 5% of the entire dataset). In order to simulate an actual adversarial attack, unlabeled sets or victim nodes tend to have higher degrees, for which we select the nodes with degrees greater than 10.

3.5.2 Experiment Setting

To verify the model accuracy and robustness, we compare our PGGCN framework with the state-of-the-art models in clean and attacked graph data. These four baseline GCN models includes:

- The original GCN model (GCN) [45].
- Deepcloak model (DC) that aims at minimize the feature differential [25].
- Gaussian-based Robust GCN model (RGCN) without the pairwise differential [110].
- Adversarial training model (AT) that injects the adversarial samples into the training set to classify future adversarial examples correctly [83] .

We also conduct three types of widespread adversarial attacks:

Table 3.2: Prediction accuracy of all models in each dataset

		Raw Data					Nettack				
		GCN	DC	RGCN	AT	PGGCN	GCN	DC	RGCN	AT	PGGCN
Accuracy (%)	Cora	80.3	78.6	77.3	80.1	78.4	6	43.1	63.2	45.1	67.1
	Citeseer	74.3	72.2	68.6	74.1	73.1	16.6	45.3	57.2	42.2	60.5
	Pubmed	78.2	75.4	75.4	78.4	77.8	22	40.6	63.4	51.2	66.8
		RL-S2V					Random Attack				
		GCN	DC	RGCN	AT	PGGCN	GCN	DC	RGCN	AT	PGGCN
Accuracy (%)	Cora	21.1	45.3	65.1	50.1	70.2	49.6	47.3	69.3	57.7	73.5
	Citeseer	25.2	46.2	58.2	51.2	64.2	45.5	44.6	60.6	54.2	66.2
	Pubmed	23.4	44.2	64.2	50.2	65.2	50.8	49.2	68.2	52.5	71.3

- Nettack that generates adversarial perturbations by searching the perturbation space [114].
- RL-S2V that is a reinforcement approach of implementing adversarial graph attack [19].
- Random attack that randomly adds/deletes edges among nodes.

We used ReLU as non-linear activation for all GCN and Adam as optimizer. The learning rate is 0.01, and the hyperparameter λ is set as 0.07. The analysis of hyperparameter can be found in Section 3.5.4.

3.5.3 Experiment Analysis

Table 3.2 shows the prediction accuracy of all models. On the raw data without an attack, the classification accuracy of PGGCN is slightly lower than GCN and Adversarial Training, around 2%, due to the use of Gaussian GCN as the main framework that improves robustness but introduces random noise from sampling, which affects the classification accuracy. However, PGGCN's classification accuracy is higher than that of RGCN. This is because PGGCN adopts a pairwise structure where the difference loss ensures robustness while lim-

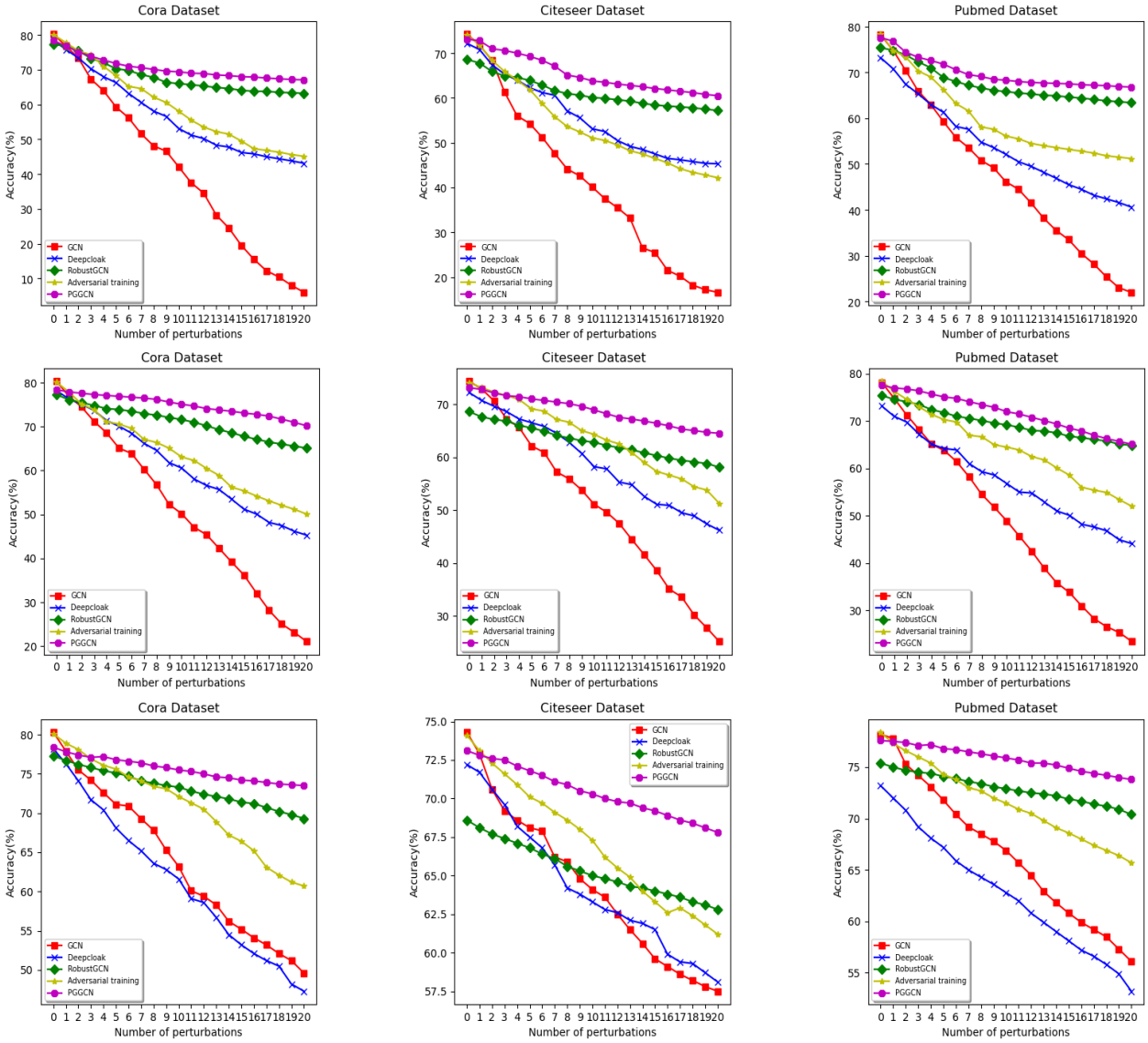


Figure 3.4: Prediction accuracy for all models under various adversarial attack

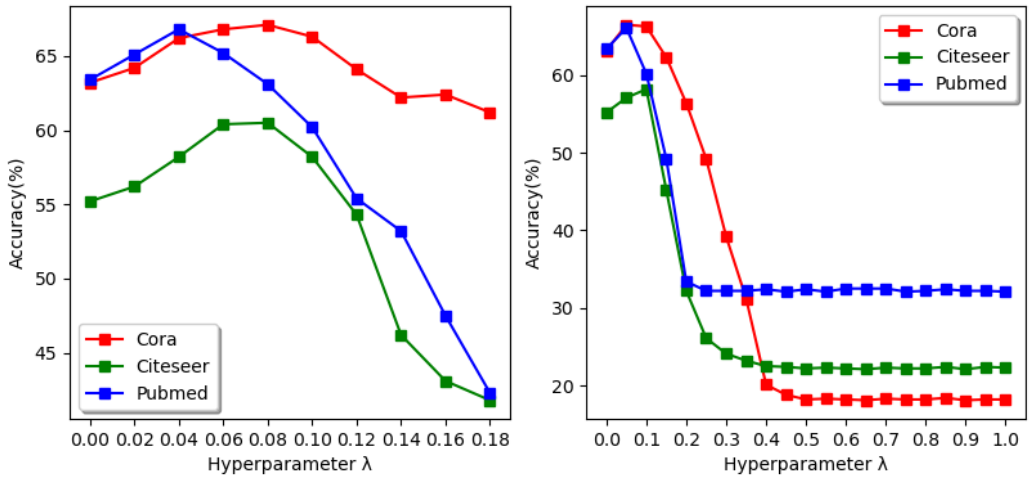
iting the randomness of the Gaussian GCN features. Therefore, PGGCN can maintain a high classification accuracy on the raw datasets.

As shown in Table 3.2, our PGGCN model outperforms all the baseline models under three different types of adversarial attacks on all datasets. PGGCN's classification accuracy

is significantly better than that of the original GCN, indicating its better robustness against adversarial attacks. Additionally, as graph data usually shares some intrinsic properties (such as features of two neighboring nodes tend to be similar), masking these key features in Deepcloak (DC) or minimizing the difference feature distance in Adversarial Learning (AT) may destroy these intrinsic properties and reduce classification accuracy. PGGCN, on the other hand, uses Gaussian distribution to absorb the noise caused by the adversarial attack and preserve these intrinsic properties. As a result, PGGCN's classification accuracy is much better than Deepcloak and Adversarial Training. Compared to RGCN, PGGCN uses a pairwise structure to reduce the randomness of classification caused by Gaussian distribution sampling, making PGGCN better than RGCN in terms of classification accuracy.

Fig. 3.4 illustrates the impact of perturbations on the classification accuracy of our PGGCN under Nettack, RL-S2V, and Random Attack. As the number of perturbations increases, the noise from the adversarial attacks becomes more significant, leading to a decrease in the model's classification accuracy. The slope of the curves in Fig. 3.4 can represent the robustness of each model. RGCN and PGGCN exhibit stronger robustness than the other models as the model accuracy of RGCN and PGGCN decreases less with increasing perturbations. This experimental result suggests that the Gaussian structure can improve model robustness better than directly masking key features in Deepcloak (DC) or minimizing the difference features in Adversarial Learning (AT). Therefore, we can conclude that the pairwise structure of our PGGCN can help reduce the classification error, improve the classification accuracy, and ensure the model robustness.

3.5.4 Fidelity-Robustness Analysis



(a) λ from 0 to 0.2 (b) λ from 0 to 1

Figure 3.5: Impact of hyper-parameter λ on prediction accuracy

In order to demonstrate the trade-off between fidelity and robustness of thIS model, we design experiments to verify the effect of hyper-parametric $\lambda \in [0, 1]$ on the classification accuracy of PGGCN, where it classifies the data received after Nettack. According to the Eq. (3.25), the hyperparameter λ is used to balance the classification loss and distance loss. When λ is 0, the model only calculates the classfication loss, in which case the model has the greatest fidelity. When λ is 1, the model only calculates the distance loss, in which case the model has the greatest robustness.

According to Fig. 3.5, with the growth of λ, the classification accuracy increases to an appropriate value first and then decreases. In particular, when $\lambda = 0$, PGGCN model only considers classification loss and is essentially a Gaussian GCN in this case. As the value of λ increases, the weight of differential loss in the PGGCN model increases, and the

weight of the difference loss decrease, which can enhance the robustness and the classification accuracy of PGGCN. When λ is higher than such an appropriate value, PGGCN considers the mode robustness more than the overall classification accuracy, so that the classification error becomes more significant.

3.6 Conclusion

In this part, to resist adversarial attack, we propose a novel graph convolution model, PGGCN, to enhance the robustness of GCN. In PGGCN, by using Gaussian distribution in hidden layers of the Gaussian GCN to obtain graph features and perturbed graph data in the Perturbed GCN to get adversarial graph features, our proposed method PGGCN can effectively alleviate the impact of adversarial attack on node classification performance. The balance between robustness and fdelity is also ensured by analyzing the lambda values in the loss function. Experimental results demonstrate that our proposed method can improve the robustness of GCN while maintaining classification accuracy under various adversarial attack strategies.

CHAPTER 4
DATA TRAINING FRAMEWORK

4.1 Introduction

Maching learning training requires a large amount of data collected by many institutions. When those insitutions cooperate and share data, users's privacy can be seriously violated [8][7][10][105][12]. Researchers have proposed a distributed machine learning framework called federated learning (FL) to solve the privacy protection problem in this scenario[55][91]. FL framework uses a central server to help multiple devices collaborate to train a global model. Specifically, the client of FL is responsible for data storage and local model training, while the server is responsible for collecting the local model and aggregating a global model. The FL framework allows joint training without data sharing to meet the requirements of privacy protection. In recent years, federated learning has been applied to many fields, such as autonomous vehicles [38][95], recommendation systems [74][108], healthcare [99][30],Blockchain [79][78], and IOT [92][32].

Although FL significantly increases privacy protection in machine learning training, privacy leakage is still possible to malicious attackers who can obtain the model parameters transmitted by the clients. For example, by using the membership inference attack [36], attackers can determine whether a data point exists in the client; by applying the the model inversion attack [101], attackers can restore the training data of a trained model through the model gradient. Furthermore, malicious servers are able to launch a gradient attack to restore privacy clients data via their access to the client models, which fudamentally

undermines the privacy protction capability of federated learning.

An intuitive way to defend against the gradiant attacks is to remove the central server that is able to obtain parameters and gradients from the clients. [105][66] Following this idea, we propose the Decentralized Federated Learning Framework (DEFEAT). The DEFEAT framework does not rely on clients sharing privacy data or aggregating client parameters with a central server. Instead, DEFEAT framework uses a peer-to-peer (P2P) network structure where each independent client interconnects with neighboring clients. Therefore, during the training process, the clients will only communicate their parameters with one-hop neighbors. Specifically, clients train the local model with their own data in the model training phase. Then, in the communication phase, the clients send the local model to their neighboring clients within one hop through the peer-to-peer network. Finally, during the aggregation phase, clients update their new local models by aggregating the received model from their neighbors. Thus, each client trains and aggregates a personalized model due to the difference in neighbors owned by the client. Furthermore, the DEFEAT framework does not have a global model, which means that a malicious attacker cannot infer the training gradients of other models from the global model parameters, and protecting the data privacy. To balance the communication cost of P2P and the training accuracy of the model, we design several training schemes to ensure that DEFEAT achieves proper model training with communication efficiency.

The contributions of the paper are as follows:

- We propose a novel framework for Decentralized Federated Learning called DEFEAT

to resist gradient attacks. The DEFEAT framework relies on a P2P network to transfer model parameters among clients and jointly train client personalized models without a global model.

- We investigate the gradient attacks in FL and study how to resist gradient attacks by analyzing the training process of centralized and decentralized federated learning frameworks. We further discuss why our DEFEAT model can prevent client side data leakage.

- We design a series of experiments to evaluate the DEFEAT framework proposed in this paper. We compare the DEFEAT framework with other state-of-the-art frameworks for model training accuracy and measure the privacy-preserving ability of DEFEAT for different gradient attacks.

The rest of this paper is organized as follows. The current studies in related fields are introduced in Sec 4.2. The preliminary is introduced in Sec. 4.3, following which we propose our DEFEAT framework in Sec. 4.4. The experimental results and analysis are provided in Sec. 4.5. Finally, we summarize this article in Sec. 4.6.

4.2 Related Work

In this section, we present the background of gradient attacks, and introduce the development of decentralized federated learning at current stage.

4.2.1 Gradient Attacks

The federated learning framework enables individual clients to train a deep model jointly without the data leaving the local area. The clients train the local model parameters with their own data, and a central server collects and aggregates these local parameters to generate a new model. This process is considered to be secure privacy-preserving training. However, recent studies have shown that a malicious server can effectively restore the local dataset through the local and global parameters, which significantly affects the privacy-preserving capability of federated learning. Le *et al.* [49] were the first to find that clients' private data can be restored by the gradient information, e.g. the training gradient can be used to determine whether a sample or label exist in the training dataset. Hitaj *et al.* [33] proposed a GAN-based data restored network, which uses the global model as the parameters of the discriminator and merges the generated samples into the training set to train the classifier to efficiently generates private data. Zhu *et al.* [111] proposed the deep leakage of gradient (DLG) framework. The DLG method generated dummy gradients by inputting dummy data and dummy labels, and restored the private data and labels by optimizing the dummy gradients. Zhao *et al.* [103] added a label restore module to DLG, which significantly improves data restore speed by predicting the labels presented in the data. In the work of [26], the authors stated that even the multi-image federated learning does not guarantee the privacy of all user data. The images can be restored in each batch of one hundred images. Yin *et al.* [101] proposed the GradInversion method to recover the private image by converting the given batch average gradient. These attacks significantly compromise the security of federated

learning. Although gradients can be protected by certain methods such as differential privacy or homomorphic encryption, they cannot essentially solve the problem of privacy leakage.

4.2.2 Decentralized Federated Learning

Currently, there are two research directions in decentralized federated learning. One relies on election mechanisms to select a node from the peerto-peer (P2P) network as a temporary central server to ensure the fairness and security of the network. Behera *et al.* [3] proposed a decentralized federated learning framework based on RAFT selector, which has no central server but achieves aggregation of models by continuously selecting temporary nodes as central servers. Wang *et al.* [84] proposed a novel federated framework called "swarm learning", that combined edge computing and blockchain-based peer-to-peer networks. During each iteration of model training, a node is dynamically selected as the server to avoid centralization of power and ensuring a fair distribution of control. The other direction is to obtain global consensus through the communications among clients. Lalitha *et al.* [48] propose a fully decentralized federated learning framework in which users train the framework using a Bayesian-like approach and introduce beliefs on the model parameter space. Based on this work, Lalitha *et al.* [47] futher proposed an improvement framework, where nodes learn models by aggregating one-hop neighbor data of their local data and by continuously updating the global beliefs of the models. Hu *et al.* [34] proposed a gossip-based decentralized federated learning method. In this framework, the model parameters are transmitted through the peer-to-peer network in segments using the gossip protocol. However, each client in this framework accumulates the same model parameters, resulting in the entire framework

training the same model. Therefore, the proposed model is vulnerable to gradient attacks, as an attacker can obtain this model from any client. For the above framework, federated learning still exists temporal central server or global model, and the attacker can still infer the training gradient of the victim client through the global model to achieve the gradient reversal attack.

4.3 Preliminaries

Federated learning is a distributed machine learning framework. This framework solves the problem of data silos by jointly training machine learning models through server aggregation of client model parameters without relying on private data sharing. The most popular federated learning algorithm is FedAvg [63], shown in Fig. 4.1. Suppose there are K clients

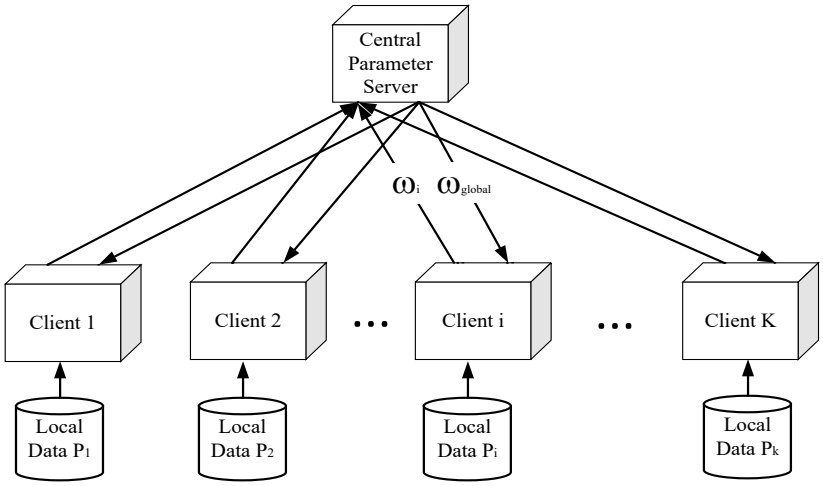


Figure 4.1: Federated Learning Framework Structure

training a machine learning model together. FedAvg distributes the global model ω^t_{global} to each client $i \in \{1, 2, \cdots, K\}$ at each training round t. The client sets the initial state of the

local model to the obtained global model $\omega_i^{t_0} = \omega_{global}^t$ In each epoch e of local training, the model picks data points (x_i, y_i) to obtainthe model gradient:

$$\nabla\omega_i^{t_e} = \frac{\partial\ell\left(F\left(x_i, \omega_i^{t_e}\right), y_i\right)}{\partial\omega_i^{t_e}}, \tag{4.1}$$

where $F(\cdot)$ is the predicted result of the model for the input data x_i, $\omega_i^{t_e}$ is the local model in e-th epoch and t-th training round, and $\ell(\cdot)$ is the loss function.

At each epoch e, the model updates the local model using stochastic gradient descent (SGD) as:

$$\omega_i^{t_{e+1}} = \omega_i^{t_e} - \eta\nabla\omega_i^{t_e}, \tag{4.2}$$

where η is the learning rate. The local training will eventually go through E epochs, and the final local model is $\omega_i^{t+1} = \omega_i^{t_E}$. The central parameter server obtains the final model parameters ω_i^{t+1} for client i and aggregates all client parameters based on the data to obtain the global model ω_{global}^{t+1} for this training round:

$$\omega_{global}^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n}\omega_i^{t+1}. \tag{4.3}$$

where n_i is the amount of data for the client i and n is the amount of data for all clients.

During the FedAvg process, only the local model and global model parameters are transmitted between the server and clients, achieving the joint training of multiple clients without sharing local data. However, recent research has shown local private data can be leaked by the gradients sent from clients to the server. Among many gradient attacks, Zhu et al. proposed that the Deep Leakage from Gradients(DLG) model [111] can significantly compromise federated learning security.

The core idea of DLG is to invert the training data by obtaining the model gradient from each update. Specifically, DLG assumes a fake data point (x'_i, y'_i) in the beginning. By feeding the data point into the global model ω^t_{global} in training round t, attackers can obtain a fake model gradient$\nabla\omega'^t_i$:

$$\nabla\omega'^t_i = \frac{\partial\ell\left(F\left(x'_i, \omega^t_{global}\right), y'_i\right)}{\partial\omega^t_{global}}. \tag{4.4}$$

According to Eq. (4.2), relying on the global model $\omega^{t_0}_i = \omega^t_{global}$ and the updated local model $\omega^{t_E}_i$, the true model gradient can be calculated as:

$$\nabla\omega^t_i = \frac{\omega^{t_0}_i - \omega^{t_E}_i}{\eta} \tag{4.5}$$

Then, DLG updates the fake data point (x'_i, y'_i) closer to the true data point (x_i, y_i) by minimizing the distance between the spurious model gradients and the true model gradient. Thus the target equation of DLG is:

$$\begin{aligned} x'^*_i, y'^*_i &= \arg\min_{x'_i, y'_i} \left\|\nabla\omega'^t_i - \nabla\omega^t_i\right\|^2 \\ &= \arg\min_{x'_i, y'_i} \left\| \frac{\partial\ell\left(F\left(x'_i, \omega^t_{global}\right), y'_i\right)}{\partial\omega^t_{global}} - \frac{\omega^{t_0}_i - \omega^{t_E}_i}{\eta} \right\|^2 \end{aligned} \tag{4.6}$$

where x'^*_i, y'^*_i are the recovered data and label. It should be noted that this gradient descent algorithm requires the computation of the second-order gradient. Therefore this algorithm requires the prediction function $F(\cdot)$ to be second-order derivable, which is achievable for most machine learning models.

4.4 Methodology

4.4.1 Peer to Peer Network

DEFEAT relies on peer-to-peer (P2P) networks for communication between clients and enables fully decentralized federated learning framework. This P2P network is represented by the graph $G = (V, E)$, where V represents the set of all clients and E represents the set of all communications between clients. For any client $i \in V$, $N(i)$ is defined to be the set of neighboring clients, that is, the set of all clients for which there exists communication $(i, j) \in E$.

In DEFEAT, client i communicates periodically with its neighbor node $N(i)$. For each communication round, i can obtain the model parameters of one hop client in the P2P network, and at least q rounds of communication are required to obtain the parameters of the farthest q-hop client. At the same time, the average number of neighbors $R = N(i)$ of each node affects the amount of communication data in each communication round. Therefore, balancing the average path length Q and the average number of neighbors R of each node is a key focus when building a P2P network. The relationship between the Q and R can be calculated using the theory of six degrees of separation [85], which is calculated as the following equation:

$$Q = \frac{\log(K)}{\log(R)}, \tag{4.7}$$

where K is the number of clients, R is the average of neighbors, and Q is the average path length. According to Eq. (4.7), the number of neighboring nodes R needs to be adjusted depending on the total number of clients K. When the number of neighbor nodes R is too high, the cost of communication goes up and can lead to a communication overheating.

When the average number of hops Q is too high, the client needs more communication rounds to engage enough training data for training and reduces the speed of model convergence.

4.4.2 Decentralized Federated Learning Algorithm

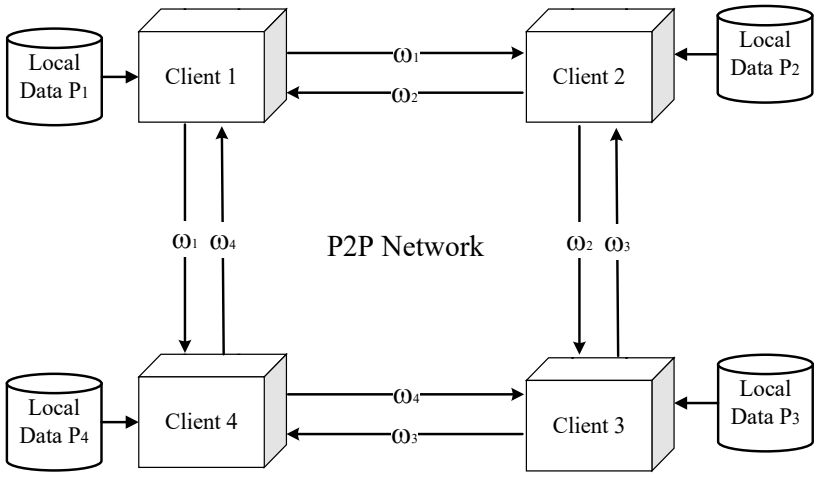


Figure 4.2: Decentrilized Federated Learning Framework Structure

The decentralized federated learning framework only has clients, as shown in Fig. 4.2. Each client i has an independent database storing local privacy data P_i. The client i are connected to their neighboring nodes and form a P2P network to transmit model parameters ω_i.

When the DEFEAT framework performs model training, it goes through three states: initialization, local training, and parameter aggregation. The initialization state is responsible for building the P2P network structure and initializing the parameters of each model. In the local training state, each client trains with the local dataset P_i. In the parameter aggregation state, each client transmits the locally trained model parameters to its neighboring

client and then aggregates the models with the received model parameters. The operation of two states: local training and parameter aggregation, are integrated and termed as a training round. In the DEFEAT framework, multiple training rounds are are used to obtain trained models. Hereafter, we introduce the three states in more details.

Initialization. The purpose of the initialization state is to build a P2P network and to initialize the model parameters. DEFEAT needs a server to assist in this state. Specifically, the server builds a suitable P2P network structure according to the number of participating clients, and the clients communicate with their assigned clients according to the P2P structure. Then the server sends the initialization model parameters to all clients to ensure the uniformity of the model initialization. Finally, the server and clients are disconnected and then clients enter the model training phase. Note that the server is only involved in the initialization state and not in the model training state, so the whole framework is a decentralized federated learning framework.

Local Training. The goal of the local training state is to update the local model based on the local dataset. For example, for client i in the framework, its local model in the t-th training round is ω_i^t. Local training assigns minibatch B as input from the local dataset P_i and performs E epochs of training. When the model is fed with local data to obtain the training gradient, stochastic gradient descent (SGD) is used to optimize the local model parameters as shown in Eq. (4.1) and Eq. (4.2). When E epochs of local training are completed, the final local model parameters are $\overline{\omega}_i^{t_0} = \omega_i^{t_E}$.

Parameter Aggregation. The parameter aggregation state performs parameter com-

munication amongst clients. Specifically, there are $D, (D \geq 1)$ communication rounds in each parameter aggregation. In each communication round $d \in \{1, 2, ..., D\}$, client i transmits the locally model parameters $\overline{\omega}_i^{t_d}$ to all neighbors and receives all neighboring model parameters. After the transmission, client i aggregates all received parameters with the local parameter, this process can be expressed as:

$$\overline{\omega}_i^{t_{d+1}} \leftarrow \frac{1}{|N(i)|} \sum_{j=1}^{|N(i)|} \overline{\omega}_j^{t_d}, \tag{4.8}$$

where $j \in N(i)$ represent the neighboring client of client i, d is the current communication round.

When D communication rounds are completed, the model moves to the next training round. The last aggregated model parameters $\overline{\omega}_i^{t_D}$ are set to the local model parameters for the next training round, where $\omega_i^{t+1} = \overline{\omega}_i^{t_D}$.

The DEFEAT algorithm is also shown in Algorithm 1.

4.4.3 Training Schema

Since decentralized federated learning does not have a central server and a global model, different topologies and training schema can affect the convergence speed and training effect of DEFEAT. Therefore, we design a series of training schemes to ensure the balance of training efficiency and speed. Specifically, we design different P2P network topologies and values of communication rounds D.

For a P2P network topology with 100 clients, we divide the network into DEFEAT-3, DEFEAT-5, and DEFEAT-7 based on the number of neighboring nodes $|N(i)|$ each node

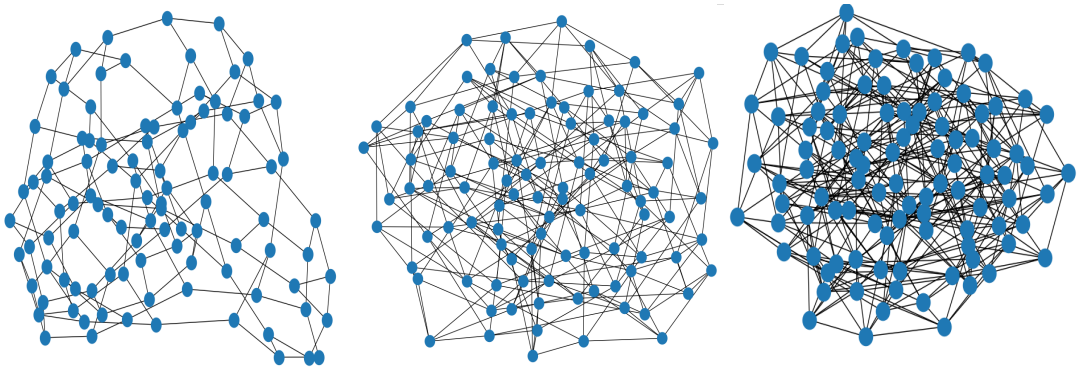
Algorithm 1 Decentralized Federated Learning Algorithm.

Inputs: Dataset P_i from each client $i \in V$.
Outputs: K well trained models ω_i^T for each client $i \in V$.
Initialization:
The server forms a P2P network and initialize the model parameters of each client ω_i^0.
for DEFEAT training round t from 0 to T **do**
  Local Training:
  $\beta \leftarrow$(split P_i into batches of size B)
  for each local epoch e from 0 to E **do**
    for batch $b \in \beta$ **do**
      $\omega_i^{t_{e+1}} = \omega_i^{t_e} - \eta \nabla \omega_i^{t_e}$
    end for
  end for
  $\overline{\omega}_i^{t_0} = \omega_i^{t_E}$

  Parameter Aggregation:
  for Communication round d from 1 to D **do**
    Send $\overline{\omega}_i^{t_d}$ to all neighborhood node $j \in N(i)$
    Receive $M = |N(i)|$ data parameters
    Updata local model:
    $\overline{\omega}_i^{t_{d+1}} \leftarrow \frac{1}{M} \sum_{j=1}^{M} \overline{\omega}_j^{t_d}$
  end for
  $\omega_i^{t+1} = \overline{\omega}_i^{t_D}$

end for

has. As shown in Fig. 4.3, More neighboring nodes in the graph represent the tighter the P2P network and the smaller the farthest distance in the network. The advantage of more neighbors is that the model converges faster with more clients involved in each training round. The disadvantage is that each training round requires passing more parameters and increasing the communication cost. Therefore, choosing an appropriate neighbor value can balance the convergence speed and the communication cost. According to Eq. (4.7), each node of DEFEAT-3 needs to send the parameters three times each communication round, and the maximum length of the graph L is 5. For DEFEAT-5, the parameters

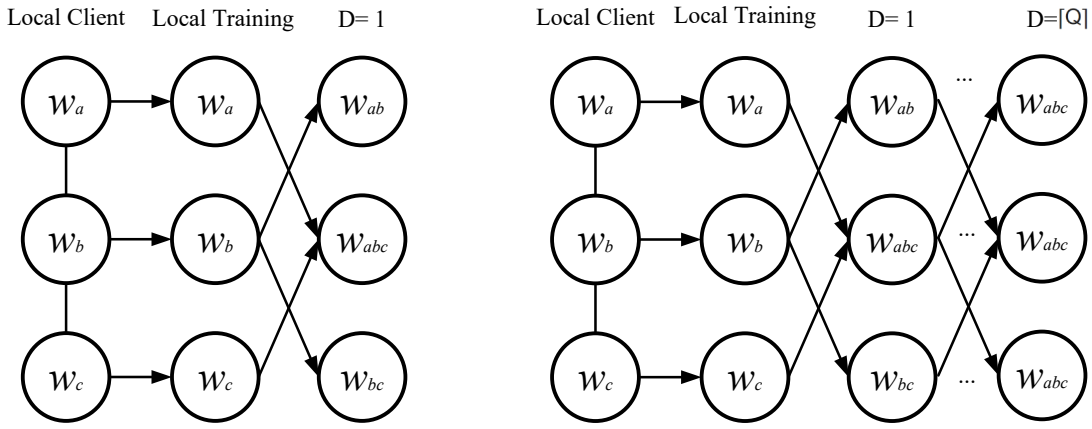


(a) DEFEAT-3 (b) DEFEAT-5 (c) DEFEAT-7

Figure 4.3: Topology of peer to peer network for decentralized federated learning

must be sent five times per communication round, and the maximum length of the graph L is 3. For DEFEAT-7, the number of communication and maximum length is 7 and 2, respectively. The denser the graph of P2P, the higher the communication cost, and the model will converge relatively faster. In contrast, the sparser the graph of P2P, the lower the communication cost, and the model will converge relatively slower. To simulate a realistic scenario of federated learning applications, we also design a DEFEAT training schema in a random topology framework DEFEAT-R. In DEFEAT-R, clients are not required to connect to adjacent clients as instructed by the server but spontaneously connect to possible clients based on the reliability of the communication. In this case, the server no longer designs the P2P topology in advance. Instead, the average path length Q and the average number of neighbors R are calculated based on the randomly generated network.

For communication rounds D, we design two different training schema, as shown in Fig. 4.4. In the first training schema, the communication rounds D equals to 1. In this



(a) schema1 (b) schema2

Figure 4.4: Different number communication rounds D in each DEFEAT training round

case, the model only communicates with nodes in one-hop. As a result, it is more difficult to obtain a global consensus. Therefore, the model is prone to oscillations, and they converge relatively slowly. However, the advantage of this schema is that the communication cost for each node is relatively small. In the second training schema, the communication rounds D equals to L. The value of L is equal to the farthest distance of the graph structure. In this case, the model prioritizes the global consensus in each training round. As a result, the model requires fewer training rounds for a single client to reach convergence, and all clients tend to train the same model. However, for each training round, the model requires more communication rounds.

4.4.4 Security Analysis

This section analyzes the attacks of Deep Leakage from Gradients (DLG) against the centralized federated learning framework and the decentralized federated learning frameworks. For the centralized federated learning framework, the central parameter server can act as the attacker and restore the local training data P_i of target client i. Whereas, for the decentralized federated learning in Fig. 4.2, any client j can act as an attacker to restore the private training data P_i of neighboring nodes $i \in N(j)$. In centralized federation learning (e.g., FedAvg), the gradient $\nabla\omega_i^t$ can be obtained from the inverse derivation of Eq. (4.2). To simplify the calculation, we assume that the epoch value $E = 1$, thus the gradient obtained by attacker in FedAvg is:

$$\nabla\omega_i^t = \frac{\omega_{global}^t - \omega_i^{t+1}}{\eta}. \tag{4.9}$$

Since the server controls the global model ω_{global}^t and all the local model ω_i^{t+1}, it is possible that the malicious server can perform data restoration to attack target clients.

In our DEFEAT framework, the core idea of resisting DLG is to hide the model gradient. According to Eq. (4.2), the gradient $\nabla\omega_i^t$ in DEFEAT can be calculated from the difference between model parameters:

$$\nabla\omega_i^t = \frac{\omega_i^t - \overline{\omega}_i^{t_0}}{\eta}. \tag{4.10}$$

However, the attacker only receive the updated model $\overline{\omega}_i^{t_0}$ in communication round. There is no way for an attacker j to get the local model ω_i^t of client i, which makes it impossible for

the attacker to obtain the exact model gradient. In a more detailed analysis, DLG works for other federated learning because all clients tend to train the same global model ω_{global}^{t}. In DEFEAT, each client trains a different local model independently, which leads to the model being unable to obtain the corresponding gradient by model parameter calculation.

4.5 Performance Evaluation

4.5.1 Datasets

In this paper, the Fashion-MNIST dataset [88] is selected as the experimental dataset, which covers 70,000 images of different garments from 10 categories. Fashion-MNIST was divided into training dataset with 60000 image data and testing dataset with 1000 image data, and each image has the dimension of 28×28×1. To represent the differences in data distribution among clients, we divided Fashion-MNIST in three different ways:

- Average division: Dividing the dataset randomly into K equal size parts, and assigning each part to a client as a local dataset.
- Non-I.I.D and balanced division: The dataset is divided into K parts, where each part has the same amount of data but different labels. This dataset division simulates a scenario where the data distribution is different across different clients.
- Non-I.I.D and imbalanced division: The dataset is divided into K parts, where each part has a different amount of data and labels. This dataset simulates the scenarios of federated learning in real-world applications [91].

4.5.2 Experiment Setting

In order to verify the performance of the decentralized federated learning framework, we compare our proposed DEFEAT and four baselines in terms of accuracy and convergence efficiency of model training.

- Centralized machine learning framework (CML): This framework combines all image data in a central server and classifies the images by a simple four-layer CNN model.
- Centralized Federated Learning framework(CFL): In this paper, the FedAvg [63] model is used as a baseline to measure the difference between centralized and decentralized federated learning.
- Decentralized federated learning baseline: In this paper, two different decentralized federated learning frameworks are chosen. The Gossip-based decentralized federated learning framework (SGossip) [34] communicates with neighboring clients multiple times in a training round, and Global Belief-based decentralized federated learning framework (GB) [47] communicates with neighboring clients only once in a training round. These two frameworks correspond to our proposed schema 1 and schema 2 training in our proposed DEFEAT.

For every framework, we selected a four-layer CNN as our classification model, featuring the following structure: (1*28*28-32*28*28-32*14*14-64*14*14-64*7*7). We employed the cross-entropy loss as our loss function. In our simulation, we had a total of 100 clients, set the learning rate to 0.01, and established 500 training rounds for all frameworks. We utilized

Table 4.1: The accuracy for image classification in the Fashin-MNIST datasets.

		Average Division	Non-I.I.D Balanced	Non-I.I.D Imbalanced
CML		92.98%		
CFL		90.54%	88.56%	88.32%
SGossip		89.21%	88.18%	87.62%
GB		87.44%	86.36%	86.19%
Schema1 (D=1)	DEFEAT-3	82.56%	76.62%	74.43%
	DEFEAT-5	85.61%	83.57%	82.19%
	DEFEAT-7	86.21%	84.32%	84.23%
	DEFEAT-R	84.67%	82.54%	80.66%
Schema2 (D=$\lceil Q \rceil$)	DEFEAT-3	85.41%	82.31%	80.12%
	DEFEAT-5	87.31%	86.62%	86.32%
	DEFEAT-7	88.41%	87.64%	87.22%
	DEFEAT-R	85.55%	84.48%	82.26%

a local batch size of 8 and 10 local epochs. As for the training approach, we set the SGD momentum to 0.5.

4.5.3 Experiment Analysis

First, we evaluate the classification accuracy of the trained classification models for images under different frameworks, and Table 4.1 shows the classification results. It can be found that the centralized machine learning (CML) framework has the most accurate classification accuracy, because the centralized framework tends to obtain better training results. Similarly, centralized federated learning (CFL) also yields good classification accuracy. For all decentralized federated learning, the gossip-based framework (SGossip) has the highest classification accuracy. This is because the SGossip framework consumes high communication costs to enable all clients to train a model jointly. In essence, the training effect of the SGossip framework is similar to that of CFL. As for our proposed DEFEAT framework, the

Table 4.2: The label restoration accuracy.

Batchsize		1	5	10	16	32
CFL		100%	97.42%	95.74%	94.32%	88.53%
SGossip		100%	95.33%	92.31%	90.76%	85.44%
GB		100%	87.48%	84.55%	81.82%	74.64%
Schema1 (D=1)	DEFEAT-3	100%	61.24%	55.44%	52.35%	42.34%
	DEFEAT-5	100%	57.63%	54.32%	50.12%	35.43%
	DEFEAT-7	100%	55.32%	50.12%	42.62%	30.22%
	DEFEAT-R	100%	54.42%	48.78%	41.71%	27.54%
Schema2 (D=$\lceil Q \rceil$)	DEFEAT-3	100%	73.11%	67.23%	64.12%	55.46%
	DEFEAT-5	100%	71.41%	66.82%	62.11%	50.88%
	DEFEAT-7	100%	68.62%	63.42%	55.19%	48.33%
	DEFEAT-R	100%	64.88%	60.14%	53.87%	45.67%

accuracy of DEFEAT-7 is higher than that of DEFEAT-5, and DEFEAT-5 is higher than that of DEFEAT-3. This may be caused by the fact that the network becomes denser as the R value rises. The denser the network structure, the more clients participate in the training each round, and the better classification results of the model training. Similarly, considering the same P2P structure, the model accuracy of schema 2 is generally higher than that of schema 1. This is because each node in schema 2 is given priority to obtain a global consensus.

Then, we attack all federated learning frameworks using DLG and examine the restoration accuracy for private labels under each framework. Table 4.2 demonstrates this accuracy. We can see that as the batch size of training increases, the label restoration accuracy keeps dropping. This is because data with batch size N has $N!$ different permutations [111], and more permutations decreases the label inference accuracy. For all federated learning, DLG can restore privacy labels more accurately for the SGossip and CFL frameworks, with relatively poor label restoration accuracy for the GB framework and the worst restoration

Table 4.3: The image restoration result.

		FFT	PSNR
CFL		0.165	12.82
SGossip		0.276	12.02
GB		0.506	9.96
Schema1 (D=1)	DEFEAT-3	1.043	4.23
	DEFEAT-5	1.132	3.76
	DEFEAT-7	1.243	3.56
	DEFEAT-R	1.301	3.22
Schema2 (D=$\lceil Q \rceil$)	DEFEAT-3	0.854	6.56
	DEFEAT-5	0.887	6.34
	DEFEAT-7	0.932	5.42
	DEFEAT-R	0.952	4.98

accuracy for our DEFEAT framework. This is because for the DEFEAT framework, the attacker cannot obtain accurate training gradients with a unified global model and thus generates errors in optimizing the false labels in Eq. (4.6). Furthermore, for DEFEAT, the label restoration accuracy is downgraded as R increases. As the number of neighboring nodes increases, the parameters of the local client are aggregated with the parameters of more neighbor clients and lead to a significant deviation of the local model from the aggregation model, which affects the gradient calculation. Moreover, the label inference attacks perform worse when schema 1 is used than schema 2. In schema 2, a client communicates multiple rounds with its neighbors so that each client tends to train the same model. Therefore, the attacker's model parameters can approximate the victim's model parameters to obtain a more accurate gradient, which increases the accuracy of the label restoration.

Meanwhile, we compare the differences between restored and original images under different federated learning frameworks using two-dimensional Fast Fourier Transform (FFT) and Peak Signal to Noise Ratio (PSNR). A higher FFT value means that the difference

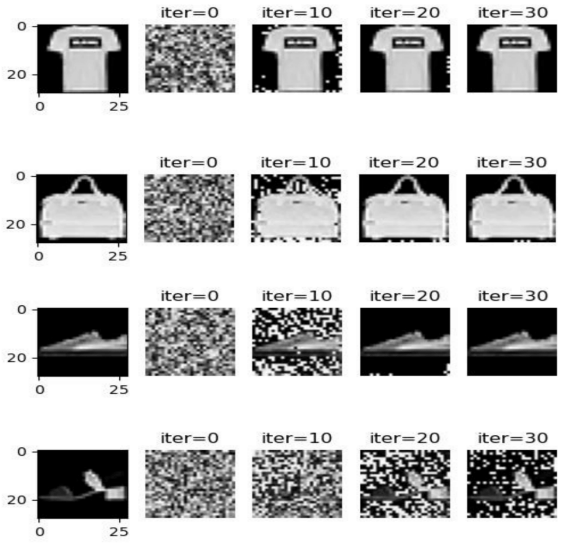


Figure 4.5: Case study of CFL result

between the restored image and the original image is larger, which means the framework is less able to protect privacy. A higher PSNR value means that the restored image by DLG is closer to the original image, representing that the framework is weaker to defend against the DLG attack. Based on the results of Table 4.3, the performance of each model on image restoration are consistent with their results on label restoration. Our DEFEAT model has higher FFT and lowers PSNR values, which means that DEFEAT is more effective against DLG attacks. This is because the attacker does not have access to the local model of the target client in the DEFEAT framework and thus cannot obtain an accurate restored image. As R increases, the privacy-preserving ability of DEFEAT is also improved because as the number of neighbor nodes increases, local parameters are aggregated with more neighboring model parameters. Thus, local parameters can be hidden better, making it more difficult for an attacker to determine the exact gradient.

Figure 4.6: Case study of DEFEAT result

Fig. 4.54.6 shows a case study of DLG image restoration for CFL and DEFEAT frameworks. The attacker separately restores four images of T-shirts, bags, sneakers, and sandals to the target client. The first column in the figure is the target image, and the subsequent columns are the images restored by the DLG model as the number of DLG training rounds increases. We find that DLG can restore the target privacy data well after 30 training rounds for the CFL framework. In comparison, for the DEFEAT framework, DLG still cannot restore the privacy data after 80 training rounds.

4.5.4 Convergence Speed and Training Efficiency

Decentralized federation learning relies on many communication rounds to enable joint client training. Therefore, we also measure the required number of communication rounds and the convergence speed in different frameworks. Table 4.4 shows the convergence speed of

Table 4.4: Training round of each model.

		Average Division	Non-I.I.D Balanced	Non-I.I.D Imbalanced
CML		18		
CFL		62	82	93
SGossip		162	184	192
GB		193	231	241
Schema1 (D=1)	DEFEAT-3	434	NC	NC
	DEFEAT-5	314	378	392
	DEFEAT-7	301	321	342
	DEFEAT-R	437	477	492
Schema2 (D=$\lceil Q \rceil$)	DEFEAT-3	323	413	451
	DEFEAT-5	265	282	292
	DEFEAT-7	164	191	202
	DEFEAT-R	391	422	440

each training framework. This experiment measures the number of training rounds when the model has converged. The results show that centralized machine learning model can converge quickly. For example, centralized federated learning (CML) can obtain a converged model in less than 100 training rounds. Decentralized learning requires more training rounds to achieve convergence of the model. Among them, the gossip-based decentralized framework (SGossip) can achieve faster model convergence. The SGossip framework communicates the parameters multiple rounds in each training round through the gossip protocol to reach a consensus of the whole network for the training model, which reduces the parameter oscillations during the model training. The global-belief-based decentralized framework (GB) guides the local model training through the global beliefs, which can also reduce model oscillations to some extent. For DEFEAT, since each client trains a local model based on local data, the model will inevitably oscillate and thus converging relatively slowly. However, as R increases, the network becomes denser, thus allowing more clients to participate in the

Table 4.5: Communication round of each model.

		Average Division	Non-I.I.D Balanced	Non-I.I.D Imbalanced
CML		18		
CFL		62	82	93
SGossip		972	1104	1152
GB		193	231	241
Schema1 (D=1)	DEFEAT-3	434	NC	NC
	DEFEAT-5	314	378	392
	DEFEAT-7	301	321	342
	DEFEAT-R	437	477	492
Schema2 (D=$\lceil Q \rceil$)	DEFEAT-3	1615	2065	2255
	DEFEAT-5	795	846	876
	DEFEAT-7	492	573	606
	DEFEAT-R	1173	1266	1320

training of DEFEAT. So, the convergence speed of DEFEAT is increasing with the increase of R. Similarly, when DEFEAT uses schema 2 for training, the local model aggregates more model parameters from non-neighboring clients in each training round. More clients are involved in the training process, and thus reducing oscillations during training and achieving fast model convergence.

In addition, Table 4.5 indicates the required communication rounds for each training framework. For federated learning, the primary time cost of training lies in communication, and the increase of communication rounds imply a decrease in the training efficiency of the model. For CFL, GB, and DEFEAT with schema 1, the number of communication rounds is equal to the number of training rounds. For SGossip framework and DEFEAT with schema 2, the frameworks employ multiple communication rounds in a training round to achieve global consensus. So, the communication rounds of SGossip framework roughly equal to the maximum value of the shortest path in the network structure multiplied by the number

of training rounds, and the communication rounds of DEFEAT in schema 2 is equal to D multiplied by the number of training rounds. In SGossip and DEFEAT with schema 2, large communication cost achieves a consensus of the decentralized framework at each training round. Furthermore, it leads to more clients participating in each training round, thus speeding up the convergence of the model and output a more accurate model. Therefore, a suitable training schema should be chosen according to the practical situation to balance the training cost and model performance.

4.6 Conclusion

This section presents DEFAT, a decentralized federated learning framework against gradient attacks. The framework abandons the central server and uses a peer-to-peer network to transmit model parameters, training a personalized model in collaboration with multiple connected clients. The DEFAT framework effectively defends against gradient attacks by preventing attackers from obtaining accurate training gradients. To balance the trade-off between training cost and model performance, we have also established various training modes to train the DEFAT framework. Finally, we compare our DEFAT with several baseline models through extensive experiments and case studies on real-world datasets. The results show that the decentralized DEFAT framework can effectively defend against gradient attacks, although it comes at the cost of increased training time and reduced model accuracy. Through the performance analysis of DEFAT framework under different modes, it can be concluded that the more decentralized the framework is, the harder it is to be attacked by

gradient, at the cost of sacrificing training time and model performance.

CHAPTER 5
DATA LEVEL RESEARCH

5.1 Introduction

Graph data is widely used in various fields, such as social networks [24], communication networks [69], and biological networks [1]. Due to privacy concerns, sensitive information in these graphs is often anonymized before being released or shared, which entails removing and/or obfuscating personal identifiers and other sensitive information [52][8][7]. But, this safeguard is not infallible. Malicious attackers can correlate anonymized graph data with external sources using sophisticated techniques, potentially leading to the unmasking of individual identities and exposure of confidential information [105]. In response, graph de-anonymization techniques have been developed to study these potential vulnerabilities as well as to improve the privacy preservation mechanisms employed in graph data sharing.

However, graph de-anonymization is fraught with challenges, particularly due to the massive scale of graph data. Traditional approaches tend to lean on exhaustive search methods to pinpoint potential matches between anonymized and query graphs. Such techniques, although comprehensive, come with computationally expensive and time-consuming, especially for large-scale graphs with millions or billions of nodes and edges [13]. Moreover, graph structures may be highly dynamic, with nodes and edges constantly added or removed. This dynamism further increases the computational complexity of the de-anonymization process. [112] Furthermore, in real-world scenarios, it is a rarity to source query graph data that perfectly mirrors its target, ushering in the need for inexact graph de-anonymization where

the query and target graphs exhibit discrepancies. These problems have become a significant roadblock for researchers and practitioners working on graph de-anonymization, limiting the applicability of these techniques to smaller graphs or necessitating the use of significant computational resources [11].

In light of the above analysis, we introduce an innovative neural-based technique specifically tailored for inexact graph de-anonymization named Graph Neural De-anonymization (GND). By acknowledging the capacity of graph vectors to maintain integral structural information, our strategy streamlines the de-anonymization process. This is achieved by transforming intricate high-dimensional graph structures into more manageable low-dimensional vector formats. Our approach is structured into three phases: the embedding phase, the comparative phase, and the matching procedure. In the embedding phase, a Graph Convolutional Network (GCN) generates embedding vectors for query and anonymized graphs, effectively capturing local and global structural information while reducing their dimensionality. The comparison phase utilizes a neural tensor network (NTN) to compare the similarity between nodes in the query and anonymized graphs. The proposed method can efficiently measure the similarity between the low-dimensional graph embeddings, identifying potential node matches based on structural and attribute similarities. Finally, the matching procedure implements a greed-based algorithm, specifically designed to identify the most suitable node pairs. By continually highlighting the most analogous node pairs and updating the similarity metrics for the yet unmatched nodes, the greedy algorithm ensures a computationally efficient approach to identifying the true correspondence between nodes in the query and

anonymized graphs. To further enhance the accuracy of the inexact matching, the proposed framework also incorporates side information as an additional source of knowledge. This side information can encompass aspects such as node attributes, properties of edges, or other pertinent contextual data, all of which can furnish invaluable insights into the intricacies of node relationships. While the graph embeddings derived from the graph convolutional network capture the structural similarity of the graphs, the side information is utilized to preserve their semantic similarity. To sum up, this paper has three major contributions:

- To the best of our knowledge, our method stands as the pioneering approach to inexact graph de-anonymization that explores the comparison of embedding vectors. By utilizing this innovative technique, we introduce a new perspective on graph de-anonymization that can lead to improved efficiency and performance compared to traditional approaches.

- We develop a novel algorithm to rapidly filter out node matches between query and anonymized graphs. This algorithm streamlines the matching process by efficiently identifying candidate node pairs with high similarity scores, significantly reducing the computational complexity associated with traditional exhaustive search techniques.

- We conduct extensive experiments to demonstrate that our neural network-based GND method is highly competitive with state-of-the-art graph de-anonymization techniques. The results shows that our neural network-based method not only achieves high accuracy but also maintains high efficiency in the graph de-anonymization process.

The rest of this paper is organized as follows. The current studies in related fields are introduced in Section 5.2. The preliminary is introduced in Section 5.3, following which we propose our GND framework in Section 5.4. The experimental results and analysis are provided in Section 5.5. Finally, we summarize this paper in Section 5.6.

5.2 Related Work

This section will introduce the traditional graph deanonymization algorithms and the neural network-based graph deanonymization methods as follow.

5.2.1 Traditional graph de-anonymisation algorithm

Graph de-anonymization is the process of correlating anonymous network structures with public datasets to determine node identities. It can be classified into two methods: seed-based and seed-free, depending on initial node information. This technique has the potential to expose confidential data. Narayanan et al. [64] first introduced a seed-based approach to de-anonymize large-scale networks. They observed that k-clique patterns are commonly found in both anoymized and query networks [80], with nodes exhibiting these patterns often representing the same entities. Given this insight, attacks can use these patterns to identify the seed nodes. Based on this apporach, Korula et al. [46] further proposed a method where the threshold is set dynamically based on the features, the threshold ensures that nodes with a higher degree have a higher likelihood of accurate identification. In this method, only nodes that exceed the threshold value in degree are considered for matching. Beyond methods based on k-clique patterns, Ji et al. [41] introduced a similarity-based matching

algorithm. In their approach, nodes are chosen from the already matched set during each iteration. The similarity between all pairs of nodes within a one-hop distance is then calculated. These nodes are subsequently incorporated into the matched set based on their results and play a role in the similarity computation for unmatched nodes. In summary, seed-based techniques offer accurate and efficient graph de-anonymization. However, obtaining seeds from anonymized graphs in real-world applications proves challenging, which constrains the use of these methods. Therefore, the seed-free method addresses the inherent challenges of extracting seeds from anonymized graphs. Pedarsani et al. [68] first use the Bayesian framework in evaluating the likelihood of accurate node mapping. By employing node features like degree and distances to neighboring nodes as fingerprints, they facilitated seed-free graph de-anonymization. Additionally, Ji et al. [39] introduced the single-phase cold start optimization based de-anonymisation (ODA) algorithm by quantifying the differences between perfect and(1-ϵ) perfect graph data. Ji et al. [40] futher introduced an innovative algorithm specifically designed to compute the similarity between any two node pairs. This method uses metrics such as node degrees, k-referenced distances, and proximity centrality, among other structural factors. Regardless, these seed-free methods aim to compensate for the reduced accuracy due to the absence of seeds by assessing node similarity from a broader perspective. While these approaches substantially have a high model complexity and often struggle to achieve a high accuracy.

5.2.2 Graph de-anonymization based on neural networks

Compared to traditional graph de-anonymization techniques, neural-based Graph de-anonymization significantly reduces the model complexity. Li et al. [51] first utilize a combination of deep neural networks and adversarial frameworks to achieve node matching. The method first anchoring the most congruent node pairs within the latent space, and activate a propagation mechanism to de-anonymize the entirety of remaining nodes. This method transitioning from local to global perspectives, echoes traditional graph de-anonymization approaches. While it decrease the model complexity, but has a low accuracy. Lou et al. [56] first proposed transforming graph structures into vector representations using graph neural networks. Through vector feature comparisons, their approach aimed at effective graph de-anonymization. However, while their method excels with identical graph structures, it exhibits limitations in inexact graph de-anonymization scenarios. Tu et al. [76] introduced the Inexact Attribute Subgraph Matching technique. This approach seeks inexact subgraph matches on attributed graphs through the optimization of graph edit distance. A key component of their method is a heuristic that employs a backtracking tree search, designed to identify optimal solutions.However, the method falls short in terms of accuracy due to its limited consideration of graph structural information. Jie et al.[70] proposed I2BGNN, a model work for identity inference in blockchains. It utilizes subgraphs as inputs and correlates transaction subgraph patterns with account identities to enhance de-anonymization processes. Similarly, Zhou et al.[109] developed Ethident, a graph neural network for Ethereum blockchain, with its design centered on a hierarchical graph attention encoder complemented by contrastive self-

supervision. While both methods achieve good results, their applicability is constrained to the blockchain, making them less suitable for scenarios such as social networks and other inexactly structured graphs.

5.3 Preliminaries

Graph de-anonymization is a process that aims to reveal the true identities of nodes and their connections in anonymized graph data. In many applications, such as social networks, communication networks, and biological networks, graph data is used to represent complex relationships between entities. To protect the privacy of individuals and sensitive information, these graphs are often anonymized before being released or shared. Anonymization techniques typically involve removing or obfuscating personal identifiers and other information that may be exposed in the graph. However, malicious attackers may attempt to compromise the privacy of individuals by linking the anonymized graph data to external information, leading to the re-identification of individuals and the disclosure of sensitive information. Graph de-anonymization techniques are developed to study these potential vulnerabilities so as to improve the privacy preservation mechanisms employed in graph data sharing.

As shown in Fig. 5.1 5.2 5.3, there is an original graph $G = (V, E)$, where V is the node set and E is the edge set. In order to hide private information, anonymization methods are used to hide the identity information of the nodes as well as to remove or add some edges in the original graph to form an anonymized graph $G_A = (V_A, E_A)$. In addition, there is a

query graph $G_Q = (V_Q, E_Q)$ obtained from other sources, which has the identity information of the nodes. Graph de-anonymization refers to bringing the identity information of the query graph to the anonymized graph by matching as many nodes from the query graph to the anonymized graph as possible.

Depending on the input provided and the information used, the graph de-anonymization problem can be grouped into two main categories. The first one requires matching point pairs with known identity information as seed information, which we call seed-based graph de-anonymization. Since some matching seeds are already available, obtaining high-quality matching pairs is easier if the initial centers are correct. However, this type of method is often dependent on the reliability of the seed pairs that are very hard to be obtained. The second method does not require seed information and is called seed-free graph de-anonymization. This type of method tends to match by structural feature information of nodes (e.g., degree, subgraph, node similarity, and descriptive information). However, those features only capture local structural information of nodes. Thus, the de-anonymization performance is greatly affected, if both graphs have similar local substructures.

However, a significant issue in graph de-anonymization lies in acquiring a query graph that perfectly matches the target graph. Consider a situation where an attacker is attempting to de-anonymize Facebook's anonymized social network graph. The target graph might be immense, encompassing billions of nodes and edges. However, the attacker can typically only access a much smaller query graph, and the node and edge information between the query graph and the target graph may not be congruent. Therefore, we propose the

concept of Inexact Graph De-anonymization to address such challenges. Inexact Graph De-anonymization embraces the fact that the query and target graphs may not be perfectly aligned and aims at finding the best possible node mappings despite these discrepancies. It offers a robust and adaptable solution that can identify similarities between nodes based on structural and attribute information, even when the query graph is a partial, distorted, or otherwise imperfect representation of the target graph.

To address the complexities of Inexact Graph De-anonymization, we develop a novel method that relies heavily on the use of GraphSAGA and Neural Tensor Networks. The choice of these models forms the backbone of our proposed method, aiding us to confront the challenges posed by the discrepancies and inconsistencies between query and target graphs.

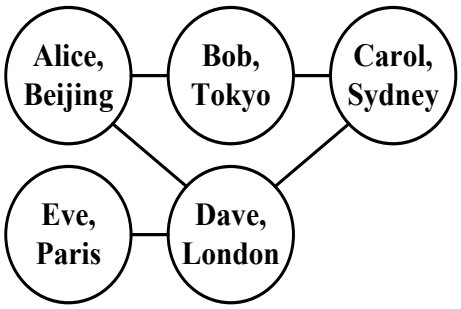


Figure 5.1: Original Graph

5.3.1 GraphSAGA

A major drawback of the traditional graph de-anonymization methods is matching inefficiency when dealing with large-scale graph as the complexity of the algorithms makes it difficult to process massive graphs in a reasonable time period. Therefore, an intuitive ap-

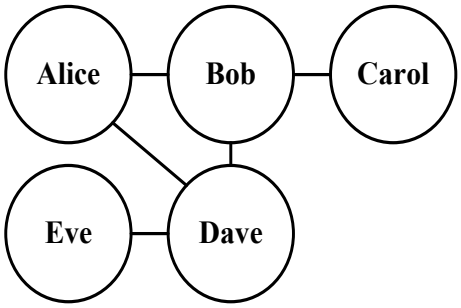


Figure 5.2: Anonymized Graph

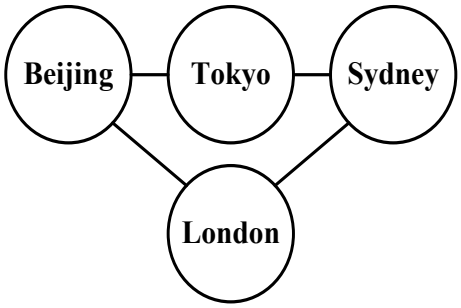


Figure 5.3: Query Graph

proach to address this issue is to adopt graph convolutional networks (GCNs) to transform the high-dimensional graph data into low-dimensional vector representations. In our proposed method, we utilize GraphSAGE, a powerful and scalable GCN, to generate graph embeddings. The core idea of GraphSAGE is to learn how to aggregate feature information from a node's neighbors and then generate a new node representation by combining the aggregated information with the node's own features.

The GraphSAGE algorithm operates through a series of steps that work together to generate graph embeddings as shown in Fig. 5.4. Initially, given a node v, a fixed-size neighborhood $N(v)$ is sampled with the size of the neighborhood determined by a parameter K that controls the number of neighbors sampled at each layer of the GCN. Following the

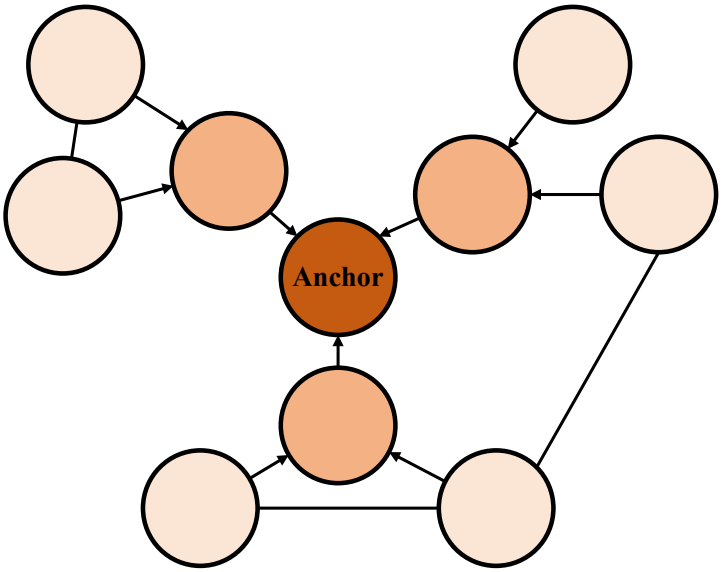


Figure 5.4: The GraphSAGE structure.

neighborhood sampling, features of the sampled neighbors are aggregated via an aggregation function as shown below:

$$h_{agg} = \text{AGGREGATE}\left(\{h_u : u \in N(v)\}\right), \tag{5.1}$$

where AGGREGATE is a common aggregation function, such as mean, max, and LSTM, and outputs a single vector h_{agg} that represents the aggregated features of the neighbors. Subsequently, the node update step combines the aggregated neighborhood feature vector h_{agg} with the node's own feature vector h_v to generate a new node representation h'_v, i.e.,

$$h'v = \text{UPDATE}(h_v, h_{agg}), \tag{5.2}$$

where the update function is referred to UPDATE, is a neural network layer such as a fully connected layer or a graph convolutional layer.

5.3.2 Neural Tensor Network

After acquiring the graph embeddings, it is crucial to find a method that measures the similarity or dissimilarity between the generated embedding vectors, which is essential for determining the correspondence between nodes in the query and anonymized graphs. Here, we introduce the Neural Tensor Network (NTN) as an efficient and powerful mechanism to compare the obtained embedding vectors. Neural Tensor Networks is presented in Fig. 5.5.

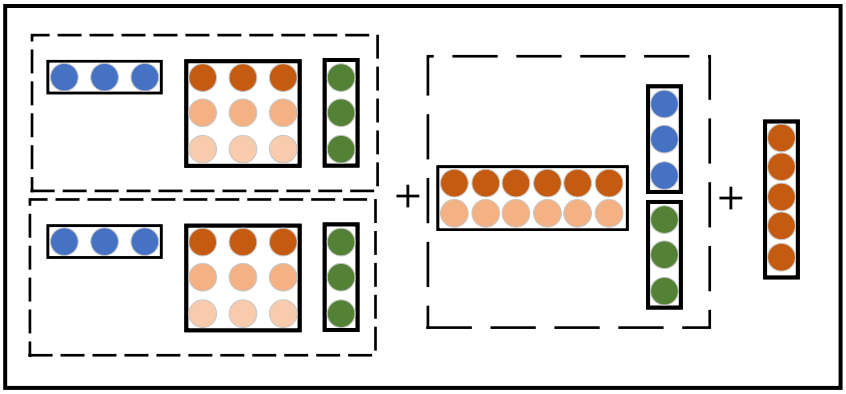


Figure 5.5: Neural tensor network structure

It is devised to capture intricate relationships between pairs of entities, such as nodes in our context. The main purpose of NTN is to learn a tensor-based scoring function that quantifies the similarity or relatedness between two input vectors. This scoring function encompasses a bilinear tensor product involving the input vectors, followed by a non-linear activation function and a linear combination of the tensor product with the input vectors as shown below:

$$f\left(h_i, h_j\right) = g\left(h_i^T W_k h_j + V\left[h_i : h_j\right] + b\right), \tag{5.3}$$

in which W_k denotes a tensor of weights, V signifies a matrix of weights, $[h_i : h_j]$ represents the concatenation of the input vectors, b is a bias vector, and g is a non-linear activation function, such as the tanh or ReLU.

5.4 Methodology

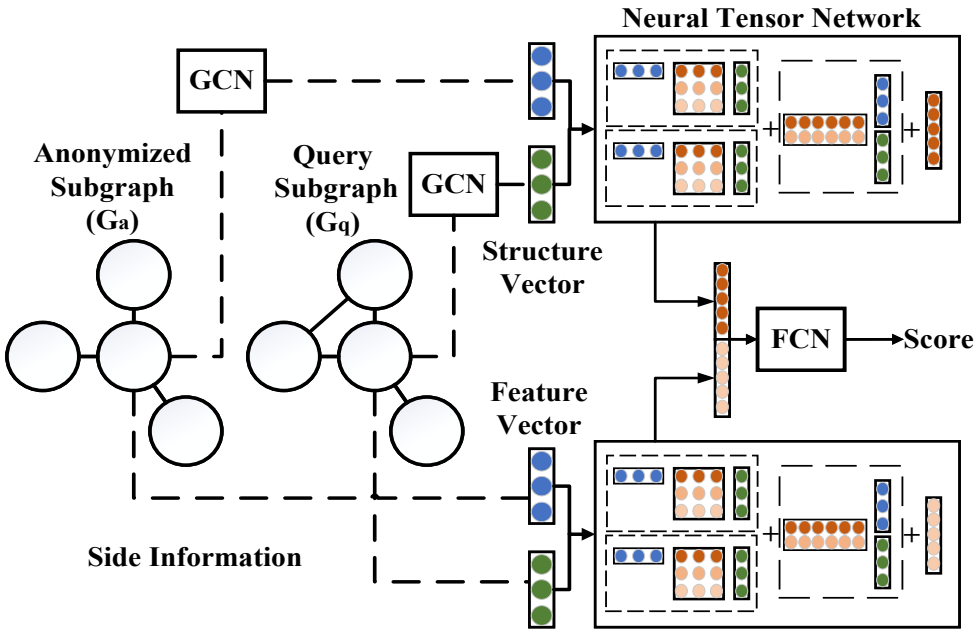


Figure 5.6: An overview illustration of Graph Neural De-anonymization

The essence of our model is the combination of node structure and side information to compute the overall node similarity, which aids in de-anonymization. This process, referred to as Graph Neural De-anonymization (GND), is illustrated in Fig. 5.6. For the structural information, our model employs a Breadth-First Search (BFS) traversal to obtain adjacent subgraphs for each node in both the anonymized graph and the query graph. Subsequently, Graph Neural Networks (GNN) are used to embed these subgraphs, which vectorize the

graph structural data. The model then compares these embedded vectors to calculate the structural similarity between every pair of nodes across the anonymized and query graphs. For node information, we compare the side information (defined as the node feature vectors) between the anonymized graph and the query graph, revealing the inherent similarity of the nodes in the two graphs.

Our graph neural de-anonymization utilizes a three-phase approach for de-anonymizing graphs:

- Embedding Phase: This is where the anonymized and query graphs are broken down into smaller overlapping subgraphs. These subgraphs are then processed and embedded using Graph Neural Networks.
- Comparison Phase: The embedded subgraphs are compared with the query subgraphs in the structural domain. while the anonymized features are matched with the query features in the feature domain.
- Matching Phase: The final phase employs a voting mechanism. Nodes with the highest frequency of matches, as informed by both structural and side information comparisons, are selected as the potential matches.

The details of our approaches are presented in the following.

5.4.1 Embedding Phase

In the embedding phase, GND decomposes the anonymous graph and the query graph into many small overlapping subgraphs and embeds them using a graph neural network. For each

node a in anonymous graph G_A and node node q in query graph G_Q, we extract a k-hop subgraph G_a around a and G_u around u via the breadth-first traversal. To ensure that we are operating within the same structural feature space, all subgraphs have an identical number of nodes. Then, the GCN maps the nodes to the embedding vectors, Z_a and Z_q. It is important to note that by using a GCN with k layers to embed the nodes, we can effectively embed the network structure of the k-hop neighborhood surrounding the central node. Therefore, embedding nodes is equivalent to embedding the whole subgraph. Accordingly, instead of comparing the embeddings of two individual nodes a and q, we are actually comparing the structure of their corresponding subgraphs vector Z_a and Z_q. During this phase, we utilize GraphSAGE [29] to perform embedding. GraphSAGE is built on the concepts of neighbor sampling and feature aggregation as shown in Fig. 5.4. It takes an input node as the anchor and samples its neighbor nodes (i.e., first-order neighbors). For each of those neighbors, we sample their neighbors (i.e., second-order neighbors) until k-order neighbor sampling is finished. In this paper, we use mean aggregator:

$$Z_v^k \leftarrow \sigma\left(W \cdot \text{MEAN}\left(Z_v^{k-1} \cup Z_u^{k-1}, \forall u \in \mathcal{N}(v)\right)\right), \tag{5.4}$$

where Z_v^k is vector for node v in the k-th layer, W is trainable weights, and σ is activation function.

5.4.2 Comparison Phase

The main idea of the comparison phase is to compute the structural distance and feature distance between the nodes in the anonymous graph and the query graph and then identifies

the nodes with the closest distance as the model outputs. After obtaining the embedding vectors of two graphs from the previous phase, a straightforward approach to model their relationship is to calculate their inner product. However, this simplistic utilization of data representations may result in inadequate or feeble interaction between the two graphs. To avoid this issue, we employ NTN [71] to measure the distance between the two graphs instead, i.e.,

$$g\left(Z_a, Z_q\right) = f\left(Z_a^T W^{[1:K]} Z_q + V \begin{bmatrix} Z_a \\ Z_q \end{bmatrix} + b\right), \tag{5.5}$$

in which W is the weight tensor, [ ] denotes the concatenation operation, V is the weight vector, b is a bias vector, and f is an activation function. K is the hyperparameter controlling the number of interaction scores produced by the model for each graph embedding pair.

In addition to the structural distance, we also compute the feature distance of the nodes based on the additional information provided by the data. We also adopt an NTN structure to measure the distance of node features.

$$g\left(F_a, F_q\right) = f\left(F_a^T W^{[1:K]} F_q + V \begin{bmatrix} F_a \\ F_q \end{bmatrix} + b\right), \tag{5.6}$$

where F_a and F_q are feature information of node a and q, respectively. After obtaining the structure similarity and feature similarity vectors, we use the concatenation operation to combine these two feature vectors. Then, we use a standard multilayer fully connected neural network to gradually reduce the dimension of the similarity score vector.

To train this model, we adopt the max margin loss method, in which the output of

positive samples should be as high as possible, while the output of negative samples should be as low as possible. Therefore, the following loss function is obtained:

$$\mathcal{L}\left(G_q, G_a\right) = \sum_{(G_q,G_a)\in P} S\left(G_q, G_a\right) + \sum_{(G_q,G_a)\in N} \max\left\{0, \alpha - S\left(G_q, G_a\right)\right\} + \lambda\|\Omega\|_2^2. \tag{5.7}$$

Here, node q and a are matching pair, P denotes the set of positive sample, N denotes the set of negative sample, G_q and G_a are two input graphs, $S(\cdot,\cdot)$ denotes the output similarity score vector after our GND model, and Ω is a regularization term used to limit the size of the parameters.

5.4.3 Matching Procedure

The matching procedure is used to increase the confidence of matched pairs by taking into account the presence of other matched pairs in the vicinity of the current matched pairs. The pseudocode of matching algorithm is presented in Algorithm 1. The basic concept behind this mechanism is to initially eliminate potential node pairs based on structural features, and then to further narrow down the remaining candidates by utilizing additional features. This approach reduces the amount of computation required by the model, and only nodes that pass through this two-fold screening process are successfully de-anonymized. This mechanism allows the model to effectively eliminate a majority of mismatched node pairs, only retaining those that are a match.

Algorithm 2 Graph Matching Algorithm.

Inputs: anonymous graph G_A, query graph G_Q, and threshold t for violation below which we predict candidates matching node pairs
Outputs: all matching node pairs (a,q)
For each node a in G_A and each node q in G_Q, the K-hop subgraphs G_a and G_q are obtained via breadth-first traversal and embedding Z is computed for all nodes.
for Node $a \in G_A$ **do**
  $L = \min \{g(Z_a, Z_q) \mid \forall q \in G_Q\}$
  if $L < t$ do
  return pair (a, q)
end for

5.4.4 Runtime Complexity

Our graph neural de-anonymization model primarily employs two phase to calculate the similarity scores between target and query nodes. Firstly, during the embedding phase, we utilize a GCN model to derive the embedding vectors for each node. The time complexity for this phase should be $O(K(|E_T| + |E_Q|))$, where K represents the layers of the GCN and $O(|E_T| + |E_Q|)$ signifies the total number of edges in all the graphs. Secondly, in the comparison phase, we need to calculate the similarity of every pair of nodes within the target graph and the query graph, for which the time complexity is $O(|V_T|\,|V_Q|)$.

5.5 Experiments Result

5.5.1 Datasets

In this section, we evaluate our proposed graph de-anonymization method using multiple datasets, each of which offers unique characteristics and challenges that allow for comprehensive assessment.

- LastFM Asia Social Network Dataset [50]: This dataset, provided by Stanford Uni-

versity, comprises a social network of LastFM users collected from the public API in March 2020. The LastFM Asia Social Network dataset offers a real-world example of a complex and large-scale social network, with users and their interactions providing a rich set of features and structural information. Evaluating our method on this dataset allows us to demonstrate the performance and applicability of our approach in handling real-world graph data with inherent noise and complex structures.

- AIDA Knowledge Graph Dataset [2]: The AIDA dataset is a knowledge graph dataset that consists of entities and their relationships extracted from various sources, such as documents and web pages. The AIDA knowledge graph dataset provides a diverse and challenging testbed for our method, as it features a wide range of entity types and relationships as well as varying levels of sparsity and density. Evaluating our approach on this dataset helps us demonstrate the robustness and flexibility of our method in handling different types of graphs with varying characteristics.

- Randomly Generated Graph Dataset (On-The-Fly): In addition to the aforementioned real-world datasets, we also use randomly generated graph data in our evaluation. This synthetic dataset offers controlled environments where specific graph properties, such as size, density, and connectivity, can be manipulated to create a range of testing scenarios. By evaluating our method on randomly generated graphs, we can assess its performance and scalability under various conditions, as well as identify potential limitations or areas for improvement.

For above dataset, we initially possess a set of anonymous graphs serving as target graphs

and extract a set of query graphs using sampling and breath first search methods. Then we generate a set of positive samples by introducing a small amount of noise (including deleting/adding edges and adding noise to node features) and a set of negative samples by random generated. In the experiments, we set 60% of the query graphs as the training set, 15% as the validation set, and 25% as the test set. Once the model has been properly trained, we de-anonymize the nodes in the test dataset according to the model and obtain the de-anonymization accuracy. Subsequently, based on the match degree for the nodes of positive and negative samples, we can calculate the model precision, recall, and F1 score.

5.5.2 Experiment Setting

Given the lack of models concerning imprecise de-anonymization in previous research, we select five baseline models for evaluation.

- NeuroMatch [56] is a widely recognized GNN-based graph de-anonymization model by leveraging GNNs to capture structural information, the model then computes similarity scores by directly comparing these embedded vectors.

- IASM [76] is a method for graph de-anonymization that uses node features. It applies a backtracking tree search technique to find the best matches. When graphs have special attributes on their nodes or edges, IASM searches for parts in the larger graph that closely resemble the desired pattern, even if they aren't a perfect fit.

- GNDNF is one of the models derived by omitting modules from our GND model to test its de-anonymization capability in isolation. GNDNF deletes the structural part of

the GND. It solely relies on node features to compute similarity. Specifically, instead of combining $g(F_a, F_q)$ with $g(Z_a, Z_q)$ using the concatenation operation, GNDNF directly employs $g(F_a, F_q)$ to calculate the similarity score vector.

- GNDSF similar to GNDNF and excludes the node information component, focusing exclusively on structural information to compute similarity. Specifically, GNDSF employs $g(Z_a, Z_q)$ to determine the similarity score vector.

- GND/threshold model aims for a stricter evaluation of the similarities between structural and node vectors by negating the threshold. While this approach can enhance the accuracy of node matching, but also makes it more challenging to de-anonymize nodes with partial matches. Specifically, the model sets the comparison threshold of the GND to zero, where $\alpha = 0$ in GND model.

All models were trained on a single GeForce RTX 2080 GPU, while the heuristic models operated on an Intel Core i5-9400F CPU. The platform utilized for training across all models was Python 3.6.6 with PyTorch 1.2.0.

5.5.3 Performance Evaluation

Table 5.1 illustrates the accuracy of different graph de-anonymization models on three different datasets. Overall, our GND model gains the highest accuracy on all datasets with 0.7013 on LastFM, 0.6855 on AIDA, and 0.7623 on On-The-Fly. This indicates that our model performs excellently on the task of graph de-anonymization in the context of social networks (LastFM), knowledge graphs (AIDA), or randomly generated graphs (On-The-Fly). The

Table 5.1: Accuracy of graph de-anonymization on different datasets

	LastFM	AIDA	On-The-Fly
NeuroMatch	0.5434	0.5219	0.5968
IASM	0.5986	0.5453	0.6712
GNDNF	0.4575	0.3987	0.5442
GNDSF	0.5644	0.5652	0.6152
GND/threshold	0.6111	0.6043	0.6635
GND	0.7013	0.6855	0.7623

Table 5.2: Precision, Recall, and F1 values for graph de-anonymization

	LastFM			AIDA			On-The-Fly		
	P	R	F1	P	R	F1	P	R	F1
NeuroMatch	0.59	0.45	0.51	0.58	0.42	0.48	0.62	0.51	0.56
IASM	0.60	0.54	0.57	0.59	0.55	0.57	0.73	0.58	0.65
GNDNF	0.41	0.22	0.29	0.40	0.21	0.28	0.54	0.32	0.40
GNDSF	0.57	0.49	0.53	0.54	0.48	0.51	0.66	0.54	0.59
GND/threshold	0.67	0.54	0.60	0.65	0.52	0.58	0.74	0.55	0.63
GND	0.70	0.64	0.67	0.68	0.63	0.65	0.77	0.74	0.75

accuracies of NeuroMatch and IASM are lower than our GND model on all three datasets. This might be due to the GND model's consideration of both node features and structural features when computing node similarities. We can also observe that the models derived by omitting certain features from the GND model (i.e., GNDNF, GNDSF, and GND/threshold) obtain less accuracies than the complete GND model. This further verifies the importance of both node and structural features in graph de-anonymization tasks and their interplay within the GND model. In summary, these results validate the superiority of our GND model when handling graph data.

Table 5.2 reports the precision (P), recall (R), and F1 scores of different graph de-anonymization models on three datasets (LastFM, AIDA, and On-The-Fly). These metrics are used to evaluate the performance of a model from different aspects. Especially, the F1

Table 5.3: Accuracy of graph de-anonymization at different query graph sizes

	OTF-30	OTF-50	OTF-100
NeuroMatch	0.5968	0.5831	0.5648
IASM	0.6712	0.6598	0.6012
GNDNF	0.5442	0.5312	0.4879
GNDSF	0.6152	0.5988	0.5490
GND/threshold	0.6635	0.6586	0.6122
GND	0.7623	0.7588	0.7457

score is the harmonic mean of precision and recall, thus providing a balanced measure. Our GND model shows superiority in all datasets and metrics, which aligns with the results from the accuracy analysis. This further confirms the efficacy of our model. Specifically, on the LastFM dataset, the GND model reaches a precision of 0.70, a recall of 0.64, and an F1 score of 0.67. Similar trends are observed on the AIDA and On-The-Fly datasets. The importance of considering both node and structural features is highlighted by comparing GND to the simplified models. For instance, GNDNF, which only uses node features, demonstrates the lowest scores in terms of all performance metrics, implying that node features alone are insufficient for the graph de-anonymization. NeuroMatch and IASM perform subpar compared to GND. Despite their acceptable performance, it is clear that incorporating both node features and structural features in GND provides a more powerful approach. The results strongly support the effectiveness of the GND model in performing graph de-anonymization tasks across different datasets and metrics, further emphasizing the importance of both node and structural features in graph analysis.

Table 5.3 presents the accuracy of all compared graph de-anonymization models with different query graph sizes (including OTF-30, OTF-50, and OTF-100). One clear trend

across all models is that the accuracy generally decreases as the size of the query graph increases. This is an expected outcome because the complexity of the de-anonymization task is directly proportional to the size of the graph. Larger graphs imply more nodes and more connections, which increases the difficulty of correctly identifying and mapping nodes to their original identities. Our model, GND, again performs the best with all query graph sizes. Notably, even with the largest graph size (OTF-100), the GND model maintains a high accuracy of 0.7457, which indicates its strong capacity to handle large-scale graphs. IASM and NeuroMatch models also exhibit decreasing accuracy with larger graph sizes but perform worse than GND. This suggests that our model is not only more accurate but also more robust against variations in graph size. The three derived models (i.e., GNDNF, GNDSF, and GND/threshold) show similar trends, while GND/threshold outperforms GNDNF and GNDSF across all graph sizes. This is consistent with our earlier analysis on the importance of both node and structural features. These results highlight the robustness of the GND model with increased graph size, further verifying its effectiveness and scalability in graph de-anonymization tasks.

Fig. 5.7 demonstrates the running time of different graph de-anonymization models with different query graph sizes (including OTF-30, OTF-50, and OTF-100). Running time is an important factor to be considered, especially when dealing with large-scale graph. From the results, we can observe that our GND model requires more time to process graphs compared to the simplified models (i.e., GNDNF, GNDSF, GND/threshold) but less time than the traditional models (i.e., NeuroMatch and IASM). The GND model uses both node

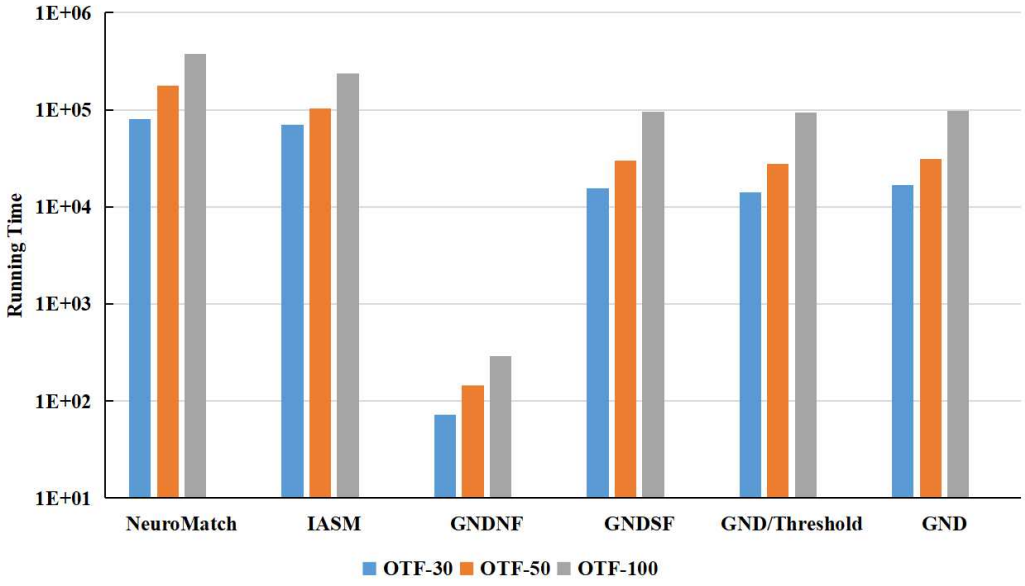


Figure 5.7: Running time of different models at different query graph sizes

and structural features, which adds complexity and hence requires additional computational time. However, its running time is relatively reasonable given its superior accuracy as observed from previous results. When comparing GND with NeuroMatch and IASM, the GND model is significantly faster. This can be attributed to the more efficient feature extraction and comparison processes implemented in GND. The substantial reduction in running time demonstrates the efficiency of our model. In terms of the simplified models, GNDNF appears to be the fastest. This is expected because GNDNF only uses node features, which reduces the complexity and hence speeds up the process. However, as shown in the previous results, the accuracy of GNDNF is much lower than the other models, indicating a trade-off between speed and performance. Therefore, our GND model achieves a good balance between performance (i.e., accuracy) and efficiency (i.e., running time).

5.6 Conclusion

This section presents DEFAT, a decentralized federated learning framework against gradient attacks. The framework abandons the central server and uses a peer-to-peer network to transmit model parameters, training a personalized model in collaboration with multiple connected clients. The DEFAT framework effectively defends against gradient attacks by preventing attackers from obtaining accurate training gradients. To balance the trade-off between training cost and model performance, we have also established various training modes to train the DEFAT framework. Finally, we compare our DEFAT with several baseline models through extensive experiments and case studies on real-world datasets. The results show that the decentralized DEFAT framework can effectively defend against gradient attacks, although it comes at the cost of increased training time and reduced model accuracy. Through the performance analysis of DEFAT framework under different modes, it can be concluded that the more decentralized the framework is, the harder it is to be attacked by gradient, at the cost of sacrificing training time and model performance.

CHAPTER 6
FUTURE RESEARCH DIRECTIONS

6.1 Federated continual learning

Social media platforms are a rich source of data. However, leveraging this data introduces significant challenges in balancing utility and privacy. Federated Continual Learning (FCL) offers a novel solution to this dilemma. FCL integrates Federated Learning and Continual Learning, presenting an approach that addresses both these concerns.

Social media data is constantly evolving. Therefore, to enhance the utility of models, we need models that can learn from new data. FCL is well-suited for this purpose. It allows for continual learning across different users, enhancing the model's utility while ensuring that user data remains private. The FCL is work as Fig. 6.1. It starts with multiple users

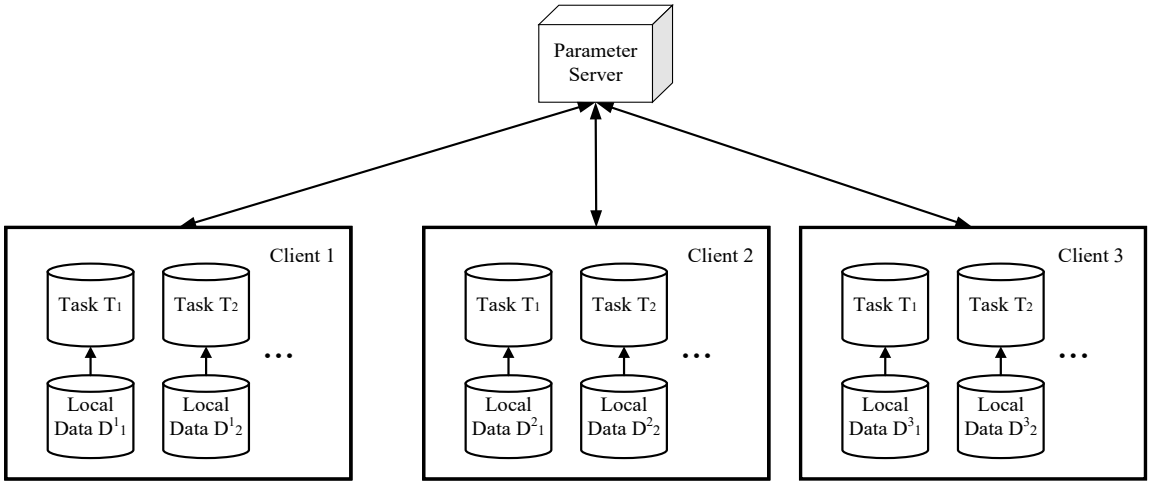


Figure 6.1: Federated Continual Learning Framework Structure

(or clients). Each client trains a model on its own specific tasks. These models then share certain information with a central server, which combines it all into a global model. For

each client, it's like a continuous learning process - they use the global model and their new data to train. But when we look at all clients together, it's a group effort, where they all contribute to the learning process without sharing their raw data. Imagine many hospitals wanting to create a shared medical model. Each hospital's data alone might not be enough. But by using FCL, they can all contribute to the model without sharing sensitive patient details. This is especially useful when medical conditions change quickly. Each hospital only keeps data for a short time, but they can still use it to update the main model.

However, FCL does face challenges. One major challenge is that data can be very different across time and places. For example, one client's data might look different from month to month. A method called Elastic Weight Consolidation (EWC) helps by tweaking only certain parts of the model while keeping the main parts stable. Another challenge is when clients can't talk to each other directly. Here, the solution is to pick the best client for a specific task based on the data they have.

Finally, it's crucial to check how well FCL is doing. We look at things like accuracy (how correct the model is), stability (how consistent it is), and plasticity (how adaptable it is). By looking at these metrics, we can understand the model's performance over various tasks.

In conclusion, as we dive deeper into using social media data, Federated Continual Learning stands out as a promising tool. It points towards a future where we can make the most of the data while keeping user privacy intact.

6.2 Personalised Decentralised Federated Learning

According to Chapter 4, we can see that the privacy and utility of the training framework for social network data is a trade-off. A decentralized federated learning ensures that each individual client has equal rights and makes it difficult for malicious attackers to target a specific victim. However, decentralized federated learning also incurs a high communication cost and may result in a loss of model accuracy.

In a decentralized approach, there is no single global model, but rather, each client trains together with a few neighboring clients. In order to train with more datas, the model requires more training rounds and more communication rounds to communicate with more clients. This is actually based on a local optimum that is centered around a portion of the data from each client. However, in real-world environments, the data from each client is non-I.I.D, which leads to a deviation between the model trained by each client and the theoretically globally optimal model. Furthermore, each training session may have a tendency to train in the opposite direction of the global optimum, resulting in the oscillation in the model training. These factors can result in the need for more training rounds in decentralized federated learning, and a high likelihood of getting stuck in a local optimum.

However, from another perspective, do we really need all participating clients to train the same globally optimal model? For example, if several hospitals are jointly training a model to diagnose the novel coronavirus, the data from a hospital in Africa will certainly be different from that in the United States. This is due to various factors such as race, geography, temperature, and so on. If a globally optimal model is jointly trained, it may not

be as effective as two different personalized models[73]. This is the personalization issue in federated learning.

The decentralized federated learning approach well satisfies the personalization characteristics of federated learning, which opens up a new direction. Perhaps a personalized decentralized federated learning framework can be designed that trains more locally suitable models with fewer training rounds, without pursuing global optimality. In this way, privacy can be ensured while further improving the utility of the model, achieving a simultaneous improvement in privacy and utility.

6.3 Client selection in federated learning

During the training process in centralized federated learning, only a portion of the clients participate in model training due to the large number of participating clients. Currently, most training approaches randomly select several clients to participate. This randomness leading to oscillation in the training process as it only trains in the direction of the local optimum. Therefore, is it possible to design a client selection algorithm to plan the participating clients in each training session to achieve the fastest model convergence?[18] Of course, this inevitably requires each client to contribute its own data distribution, which results in a certain degree of privacy leakage, and it is also a trade-off between privacy and utility.

CHAPTER 7
CONCLUSION

In the current digital era, the widespread presence of social media platforms and their corresponding data repositories provide deep insights into individual behaviors, societal trends, and global movements. This data has immense potential, offering insights for tailored marketing, emergency responses, trend predictions, political campaigns, and customer feedback. [113]

The vast potential of data brings an equal measure of ethical responsibility. Privacy is not just a regulatory requirement but a foundational element of trust between users and platforms. Without trust, data loses its reliability and value. [8] Thus, maintaining users' anonymity is vital to ensure data collection exercises respect individual choices.

Ethics go beyond business or user needs; they are morally binding, emphasizing that valuing privacy is not just strategic but inherently correct. [106] Striking the right balance between data utility and privacy is a delicate task.

Our research dives deep into this balance and focuses on the trade-off between privacy and utility in social media data mining [31]. Specifically, the paper analyzes how to balance privacy and utility in the model by examining two aspects: the trade-off between robustness and fidelity of the mining model and the trade-off between privacy and performance in the training framework. [54]

In terms of data mining model, this paper proposes a new structure called PGGCN, which improves the robustness of graph neural networks by introducing Gaussian noise and

pairwise structure and thus resists graph adversarial attacks. The model also guarantees the model's fidelity through a pairwise structure. Through loss function design, the model can freely choose the optimal balance between robustness and fidelity that best suits the current situation by adjusting the hyperparameters.

In terms of training framework, this paper studies the difference in privacy protection ability between centralized and decentralized federated learning frameworks when faced with gradient attacks. Specificly, this paper proposes a decentralized federated learning framework in which local clients transmit model parameters to their neighboring clients through a peer-to-peer network, jointly training a model without a central server. The decentralized framework can effectively defend against gradient attacks by removing the central parameter server, but it requires a large amount of communication cost and results in a decrease in model classification accuracy. Thus, selecting the training framework reasonably based on the scenario is an ideal solution.

This paper further investigates the privacy of social media data itself, specifically, it studies how to effectively use graph neural networks for graph anonymization of social networks. By studying privacy at the data level, it aims to find data mining models with high performance and good privacy protection.

Through the above research, we know that it is currently difficult to ensure data privacy without sacrificing any utility, but how to do so with a smaller cost or how to find a more reasonable balance is still an open question.

REFERENCES

[1] Eric Alm and Adam P Arkin. Biological networks. *Current opinion in structural biology*, 13(2):193–202, 2003.

[2] Simone Angioni, Angelo Salatino, Francesco Osborne, Diego Reforgiato Recupero, and Enrico Motta. Aida: A knowledge graph about research dynamics in academia and industry. *Quantitative Science Studies*, 2(4):1356–1398, 2021.

[3] Monik Raj Behera, Suresh Shetty, Robert Otter, et al. Federated learning using peer-to-peer network for decentralized orchestration of model weights. 2021.

[4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2014.

[6] Zhipeng Cai, Zhuojun Duan, and Wei Li. Exploiting multi-dimensional task diversity in distributed auctions for mobile crowdsensing. *IEEE Transactions on Mobile Computing*, 20(8):2576–2591, 2020.

[7] Zhipeng Cai and Zaobo He. Trading private range counting over big iot data. In *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, pages 144–153. IEEE, 2019.

[8] Zhipeng Cai, Zaobo He, Xin Guan, and Yingshu Li. Collective data-sanitization for preventing sensitive information inference attacks in social networks. *IEEE Transactions on Dependable and Secure Computing*, 15(4):577–590, 2016.

[9] Zhipeng Cai and Tuo Shi. Distributed query processing in the edge-assisted iot data monitoring system. *IEEE Internet of Things Journal*, 8(16):12679–12693, 2020.

[10] Zhipeng Cai, Zuobin Xiong, Honghui Xu, Peng Wang, Wei Li, and Yi Pan. Generative adversarial networks: A survey toward private and secure applications. *ACM Computing Surveys (CSUR)*, 54(6):1–38, 2021.

[11] Zhipeng Cai and Xu Zheng. A private and efficient mechanism for data uploading in smart cyber-physical systems. *IEEE Transactions on Network Science and Engineering*, 7(2):766–775, 2018.

[12] Zhipeng Cai, Xu Zheng, and Jinbao Wang. Efficient data trading for stable and privacy preserving histograms in internet of things. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–10. IEEE, 2021.

[13] Zhipeng Cai, Xu Zheng, Jinbao Wang, and Zaobo He. Private data trading towards range counting queries in internet of things. *IEEE Transactions on Mobile Computing*, 2022.

[14] Zhipeng Cai, Xu Zheng, and Jiguo Yu. A differential-private framework for urban traffic flows estimation via taxi companies. *IEEE Transactions on Industrial Informatics*, 15(12):6492–6499, 2019.

[15] S. Casas, C. Gulino, R. Liao, and R. Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[16] C. Chen, K. Li, S. G. Teo, X. Zou, and Z. Zeng. Gated residual recurrent graph neural networks for traffic prediction. *P AAAI Conference on Artificial Intelligence*, 33:485–492, 2019.

[17] Chuanxiu Chi, Yingjie Wang, Xiangrong Tong, Madhuri Siddula, and Zhipeng Cai. Game theory in internet of things: A survey. *IEEE Internet of Things Journal*, 9(14):12125–12146, 2021.

[18] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[19] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[20] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

[21] Suparna De, Maria Bermudez-Edo, Honghui Xu, and Zhipeng Cai. Deep generative models in the industrial internet of things: a survey. *IEEE Transactions on Industrial Informatics*, 18(9):5728–5737, 2022.

[22] Michal Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017.

[23] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[24] Luoyi Fu, Jiapeng Zhang, Shuaiqi Wang, Xinyu Wu, Xinbing Wang, and Guihai Chen. De-anonymizing social networks with overlapping community structure. *IEEE/ACM Transactions on Networking*, 28(1):360–375, 2020.

[25] Ji Gao, Beilun Wang, Zeming Lin, Weilin Xu, and Yanjun Qi. Deepcloak: Masking deep neural network models for robustness against adversarial samples. *International Conference on Learning Representations 2017*, 2017.

[26] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[27] Norjihan Abdul Ghani, Suraya Hamid, Ibrahim Abaker Targio Hashem, and Ejaz Ahmed. Social media big data analytics: A survey. *Computers in Human Behavior*, 101:417–428, 2019.

[28] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations 2015*, 2015.

[29] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[30] Zaobo He and Zhipeng Cai. Quantifying the effect of quarantine control and optimizing its cost in covid-19 pandemic. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2022.

[31] Zaobo He, Zhipeng Cai, and Jiguo Yu. Latent-data privacy preserving with customized data utility for social network data. *IEEE Transactions on Vehicular Technology*, 67(1):665–673, 2017.

[32] Zaobo He, Lintao Wang, and Zhipeng Cai. Clustered federated learning with adaptive local differential privacy on heterogeneous iot data. *IEEE Internet of Things Journal*, 2023.

[33] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017*

ACM SIGSAC conference on computer and communications security, pages 603–618, 2017.

[34] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[35] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. Hierarchical graph convolutional networks for semi-supervised node classification. *International Joint Conference on Artificial Intelligence 2019*, 2019.

[36] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 2021.

[37] Yan Huang, Yi Joy Li, and Zhipeng Cai. Security and privacy in metaverse: A comprehensive survey. *Big Data Mining and Analytics*, 6(2):234–247, 2023.

[38] Deepthi Jallepalli, Navya Chennagiri Ravikumar, Poojitha Vurtur Badarinath, Shravya Uchil, and Mahima Agumbe Suresh. Federated learning for object detection in autonomous vehicles. In *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 107–114. IEEE, 2021.

[39] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, and Raheem Beyah. Structural data deanonymization: Quantification, practice, and implications. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1040–1053, 2014.

[40] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, and Raheem Beyah. Structural data de-anonymization: Theory and practice. *IEEE/ACM Transactions on Networking*, 24(6):3523–3536, 2016.

[41] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, Jing Selena He, and Raheem Beyah. General graph data de-anonymization: From mobility traces to social networks. *ACM Transactions on Information and System Security (TISSEC)*, 18(4):1–29, 2016.

[42] Shouling Ji, Prateek Mittal, and Raheem Beyah. Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Communications Surveys & Tutorials*, 19(2):1305–1326, 2016.

[43] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74, 2020.

[44] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations 2014*, 2014.

[45] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[46] Nitish Korula and Silvio Lattanzi. An efficient reconciliation algorithm for social networks. *arXiv preprint arXiv:1307.1690*, 2013.

[47] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, 2019.

[48] Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. Fully decentralized federated learning. In *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[49] T. P. Le, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning: Revisited and enhanced. In *International Conference on Applications and Techniques in Information Security*, 2017.

[50] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20, 2016.

[51] Kaiyang Li, Guoming Lu, Guangchun Luo, and Zhipeng Cai. Seed-free graph de-anonymiztiation with adversarial learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 745–754, 2020.

[52] Kaiyang Li, Guangchun Luo, Yang Ye, Wei Li, Shihao Ji, and Zhipeng Cai. Adversarial privacy-preserving graph embedding against inference attack. *IEEE Internet of Things Journal*, 8(8):6904–6915, 2020.

[53] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2017.

[54] Yi Liang, Zhipeng Cai, Jiguo Yu, Qilong Han, and Yingshu Li. Deep learning based inference of private information using embedded sensors in smart devices. *IEEE Network*, 32(4):8–14, 2018.

[55] Yaguang Lin, Xiaoming Wang, Hongguang Ma, Liang Wang, Fei Hao, and Zhipeng Cai. An efficient approach to sharing edge knowledge in 5g-enabled industrial internet of things. *IEEE Transactions on Industrial Informatics*, 19(1):930–939, 2022.

[56] Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020.

[57] Guangxi Lu, Ling Tian, Xu Zheng, and Bei Hui. Integrating knowledge-based sparse representation for image detection. *Neurocomputing*, 442:173–183, 2021.

[58] Guangxi Lu, Zuobin Xiong, Ruinian Li, and Wei Li. Decentralized federated learning: A defense against gradient inversion attack. In *International Wireless Internet Conference*, pages 44–56. Springer, 2022.

[59] Guangxi Lu, Zuobin Xiong, Ruinian Li, Nael Mohammad, Yingshu Li, and Wei Li. Defeat: A decentralized federated learning against gradient attacks. *High-Confidence Computing*, page 100128, 2023.

[60] Guangxi Lu, Zuobin Xiong, Jing Meng, and Wei Li. Pairwise gaussian graph convolutional networks: Defense against graph adversarial attack. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 4371–4376. IEEE, 2022.

[61] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. Attacking graph convolutional networks via rewiring. *arXiv preprint arXiv:1906.03750*, 2019.

[62] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[63] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[64] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*, pages 173–187. IEEE, 2009.

[65] Maxence Noble, Aurélien Bellet, and Aymeric Dieuleveut. Differentially private federated learning on heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 10110–10145. PMLR, 2022.

[66] Junjie Pang, Yan Huang, Zhenzhen Xie, Qilong Han, and Zhipeng Cai. Realizing the heterogeneity: A self-organized federated learning framework for iot. *IEEE Internet of Things Journal*, 8(5):3088–3098, 2020.

[67] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[68] Pedram Pedarsani, Daniel R Figueiredo, and Matthias Grossglauser. A bayesian method for matching two similar graphs without seeds. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1598–1607. IEEE, 2013.

[69] Marvin E Shaw. Communication networks. In *Advances in experimental social psychology*, volume 1, pages 111–147. Elsevier, 1964.

[70] Jie Shen, Jiajun Zhou, Yunyi Xie, Shanqing Yu, and Qi Xuan. Identity inference on blockchain using graph neural network. In *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3*, pages 3–17. Springer, 2021.

[71] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013.

[72] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[73] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[74] Ben Tan, Bo Liu, Vincent Zheng, and Qiang Yang. A federated recommender system for online services. In *Fourteenth ACM Conference on Recommender Systems*, pages 579–581, 2020.

[75] Zhiqing Tang, Jiong Lou, Fuming Zhang, and Weijia Jia. Dependent task offloading for multiple jobs in edge computing. In *International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2020.

[76] Thomas K Tu, Jacob D Moorman, Dominic Yang, Qinyi Chen, and Andrea L Bertozzi. Inexact attributed subgraph matching. In *2020 IEEE international conference on big data (big data)*, pages 2575–2582. IEEE, 2020.

[77] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks, 2018.

[78] Chenyu Wang, Zhipeng Cai, and Yingshu Li. Sustainable blockchain-based digital twin management architecture for iot devices. *IEEE Internet of Things Journal*, 10(8):6535–6548, 2022.

[79] Chenyu Wang, Zhipeng Cai, Daehee Seo, and Yingshu Li. Tmeta: Trust management for the cold start of iot services with digital-twin-aided blockchain. *IEEE Internet of Things Journal*, 2023.

[80] Jinbao Wang, Zhipeng Cai, and Jiguo Yu. Achieving personalized k-anonymity-based content privacy for autonomous vehicles in cps. *IEEE Transactions on Industrial Informatics*, 16(6):4242–4251, 2019.

[81] Lijing Wang, Aniruddha Adiga, Jiangzhuo Chen, Adam Sadilek, Srinivasan Venkatramanan, and Madhav Marathe. Causalgnn: Causal-based graph neural networks for spatio-temporal epidemic forecasting. *AAAI*, 2022.

[82] Shen Wang, Zhengzhang Chen, Jingchao Ni, Xiao Yu, Zhichun Li, Haifeng Chen, and Philip S Yu. Adversarial defense framework for graph neural network. *arXiv preprint arXiv:1905.03679*, 2019.

[83] Xiaoyun Wang, Xuanqing Liu, and Cho-Jui Hsieh. Graphdefense: Towards robust graph convolutional networks. *CoRR*, 2019.

[84] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N Ahmad Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594(7862):265–270, 2021.

[85] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[86] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[87] WenNing Wu and ZhengHong Deng. The analysis of public opinion in colleges and universities oriented to wireless networks under the application of intelligent data mining. *Wireless Communications and Mobile Computing*, 2022:1–11, 2022.

[88] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[89] Zuobin Xiong, Zhipeng Cai, Qilong Han, Arwa Alrawais, and Wei Li. Adgan: Protect your location privacy in camera data of auto-driving vehicles. *IEEE Transactions on Industrial Informatics*, 17(9):6200–6210, 2020.

[90] Zuobin Xiong, Zhipeng Cai, Chunqiang Hu, Daniel Takabi, and Wei Li. Towards neural network-based communication system: Attack and defense. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[91] Zuobin Xiong, Zhipeng Cai, Daniel Takabi, and Wei Li. Privacy threat and defense for federated learning with non-iid data in aiot. *IEEE Transactions on Industrial Informatics*, 18(2):1310–1321, 2021.

[92] Zuobin Xiong, Wei Li, and Zhipeng Cai. Federated generative model on multi-source heterogeneous data in iot. *Accepted by AAAI Conference on Artificial Intelligence*, 2023.

[93] Zuobin Xiong, Wei Li, Qilong Han, and Zhipeng Cai. Privacy-preserving auto-driving: a gan-based approach to protect vehicular camera data. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 668–677. IEEE, 2019.

[94] Zuobin Xiong, Honghui Xu, Wei Li, and Zhipeng Cai. Multi-source adversarial sample attack on autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 70(3):2822–2835, 2021.

[95] Honghui Xu, Zhipeng Cai, Ruinian Li, and Wei Li. Efficient citycam-to-edge cooperative learning for vehicle counting in its. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):16600–16611, 2022.

[96] Honghui Xu, Zhipeng Cai, and Wei Li. Privacy-preserving mechanisms for multi-label image recognition. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4):1–21, 2022.

[97] Honghui Xu, Zhipeng Cai, Daniel Takabi, and Wei Li. Audio-visual autoencoding for privacy-preserving video streaming. *IEEE Internet of Things Journal*, 9(3):1749–1761, 2021.

[98] Honghui Xu, Wei Li, and Zhipeng Cai. Analysis on methods to effectively improve transfer learning performance. *Theoretical Computer Science*, 940:90–107, 2023.

[99] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, 2021.

[100] Xiaojun Xu, Yue Yu, Bo Li, Le Song, Chengfeng Liu, and Carl Gunter. Characterizing malicious edges targeting on graph neural networks. *International Conference on Learning Representations 2019*, 2019.

[101] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16337–16346, 2021.

[102] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[103] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[104] Zhongyuan Zhao, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. Distributed scheduling using graph neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4720–4724. IEEE, 2021.

[105] Xu Zheng and Zhipeng Cai. Privacy-preserved data sharing towards multiple parties in industrial iots. *IEEE Journal on Selected Areas in Communications*, 38(5):968–979, 2020.

[106] Xu Zheng, Zhipeng Cai, and Yingshu Li. Data linkage in smart internet of things systems: a consideration from a privacy perspective. *IEEE Communications Magazine*, 56(9):55–61, 2018.

[107] Xu Zheng, Guangchun Luo, and Zhipeng Cai. A fair mechanism for private data publication in online social networks. *IEEE Transactions on Network Science and Engineering*, 7(2):880–891, 2018.

[108] Xu Zheng, Ling Tian, Guangchun Luo, and Zhipeng Cai. A collaborative mechanism for private data publication in smart cities. *IEEE Internet of Things Journal*, 7(9):7883–7891, 2020.

[109] Jiajun Zhou, Chenkai Hu, Jianlei Chi, Jiajing Wu, Meng Shen, and Qi Xuan. Behavior-aware account de-anonymization on ethereum interaction graph. *IEEE Transactions on Information Forensics and Security*, 17:3433–3448, 2022.

[110] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407, 2019.

[111] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32, 2019.

[112] Saide Zhu, Zhipeng Cai, Huafu Hu, Yingshu Li, and Wei Li. zkcrowd: a hybrid blockchain-based crowdsourcing platform. *IEEE Transactions on Industrial Informatics*, 16(6):4196–4205, 2019.

[113] Saide Zhu, Wei Li, Hong Li, Ling Tian, Guangchun Luo, and Zhipeng Cai. Coin hopping attack in blockchain-based iot. *IEEE Internet of Things Journal*, 6(3):4614–4626, 2018.

[114] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.

[115] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.