Machine Learning and Privacy Protection: Attacks and Defenses

Dissertation
submitted to
Sorbonne Université
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Author:
Oualid Zari

To be defended on January 14th 2025, in front of a committee composed of:

Reviewers	Dr. Frédéric Giroire – CNRS, Inria, and Université Côte d'Azur - France Prof. Benjamin Nguyen – INSA-France
Examiners	Prof. Josep Domingo-Ferrer – Universitat Rovira i Virgili - Spain Prof. Antonio Faonio – EURECOM - France
Supervisor	Prof. Melek Önen – EURECOM - France
Co-supervisor	Dr. Ayşe Ünsal – EURECOM - France
Invited	Dr. Javier Parra-Arnau – Universitat Politècnica de Catalunya - Spain

Apprentissage Automatique et Protection de la Vie Privée : Attaques et Défenses

Thèse
soumise à
l'Université Sorbonne
en vue de l'obtention du diplôme de
Docteur en Philosophie

Auteur :
Oualid Zari

À défendre le 14 janvier 2025, devant un comité composé de :

Rapporteurs	Dr. Frédéric Giroire – CNRS, Inria, et Université Côte d'Azur - France Prof. Benjamin Nguyen – INSA-France
Examinateurs	Prof. Josep Domingo-Ferrer – Universitat Rovira i Virgili - Espagne Prof. Antonio Faonio – EURECOM - France
Directeur de thèse	Prof. Melek Önen – EURECOM - France
Co-directeur de thèse	Dr. Ayşe Ünsal – EURECOM - France
Invité	Dr. Javier Parra-Arnau – Universitat Politècnica de Catalunya - Espagne

Contents

1 Introduction **5**
- 1.1 Motivation . . . 5
- 1.2 Machine Learning Techniques and Privacy . . . 7
- 1.3 Privacy Attacks in Machine Learning . . . 8
- 1.4 Defenses Against Privacy Attacks in Machine Learning . . . 9
- 1.5 Problem Statement and Objectives . . . 10
- 1.6 Contributions and Outline . . . 12
- 1.7 List of Publications . . . 13

2 Background **15**
- 2.1 Foundations of Machine Learning . . . 16
  - 2.1.1 Principal Component Analysis (PCA) . . . 16
  - 2.1.2 Graph Neural Networks (GNNs) . . . 20
- 2.2 Federated Learning . . . 33
  - 2.2.1 Introduction to Federated Learning . . . 33
  - 2.2.2 Types of Federated Learning . . . 34
  - 2.2.3 Vertical Federated Graph Learning (VFGL) . . . 36
- 2.3 Differential Privacy . . . 38
  - 2.3.1 Formal Definitions of Differential Privacy . . . 40
  - 2.3.2 Properties of Differential Privacy . . . 42
  - 2.3.3 Mechanisms for Achieving Differential Privacy . . . 43
  - 2.3.4 Differential Privacy for Graphs . . . 45
  - 2.3.5 Differential Privacy in Machine Learning . . . 48
- 2.4 Challenges of Differential Privacy in Machine Learning . . . 50
  - 2.4.1 Privacy-Utility Trade-off . . . 50
  - 2.4.2 Parameter Selection (ε and δ) . . . 50

3 Related Work **53**
- 3.1 Privacy Attacks in Machine Learning . . . 54
- 3.2 Membership Inference Attacks . . . 55
  - 3.2.1 Foundations and Evaluation of MIA . . . 55

3.2.2 Defense Strategies for MIA . . . 57
3.2.3 Differential Privacy and MIA . . . 58
3.3 Link Inference Attacks . . . 61
3.3.1 Foundations and Evaluation of LIA . . . 61
3.3.2 Link Inference Attacks in Federated Learning . . . 63
3.3.3 Defense Strategies for Link Inference Attacks . . . 64

4 Privacy Considerations in Principal Component Analysis 67
4.1 Introduction . . . 67
4.2 Membership Inference Attack against PCA . . . 69
4.2.1 PCA Overfitting and Privacy Implications . . . 69
4.2.2 Attack Methodology . . . 71
4.2.3 Experimental Setup . . . 72
4.2.4 Results and Analysis . . . 73
4.3 Differentially Private PCA . . . 74
4.3.1 DP-PCA Approaches . . . 75
4.3.2 Experimental Evaluation . . . 77
4.4 Conclusion . . . 82

5 Node Injection Link Stealing Attack 85
5.1 Introduction . . . 85
5.2 Attack Methodology . . . 87
5.2.1 Threat Model . . . 88
5.2.2 Adversary's Goal and Knowledge . . . 88
5.2.3 Node Injection Link Stealing Attack . . . 88
5.2.4 Strategies for Generation of Malicious Node's Features 89
5.3 Experimental Setup . . . 92
5.3.1 Datasets . . . 92
5.3.2 Models . . . 93
5.3.3 Evaluation Methodology . . . 93
5.4 Results and Analysis . . . 94
5.4.1 Analysis of Malicious Feature Generation Strategies . . 95
5.4.2 Comparison with Baseline Attacks . . . 96
5.4.3 Impact of GNN Depth . . . 96
5.4.4 Discussion . . . 99
5.5 Defense Strategy . . . 100
5.5.1 One-Node-One-Edge-Level Differential Privacy . . . 100
5.5.2 LapGraph Mechanism for One-Node-One-Edge-Level DP 101
5.5.3 Evaluation of LapGraph Defense . . . 102
5.5.4 Discussion . . . 104
5.6 Conclusion . . . 104

6 Link Stealing Attacks in Vertical Federated Graph Learning **107**
6.1 Introduction . . . 107
6.2 Attack Methodology . . . 109
6.2.1 VFL system . . . 109
6.2.2 Gradient-based LIA . . . 110
6.2.3 Label-based LIA . . . 112
6.2.4 Baseline LIA . . . 114
6.3 Analytical Results for Link Inference Attacks . . . 116
6.3.1 Equivalence of Output-based and Label-based LIA . . . 122
6.3.2 Performance of Gradient-based LIA . . . 125
6.4 Experimental setup . . . 127
6.4.1 GNN Model architecture and learning setting . . . 127
6.4.2 Datasets . . . 127
6.4.3 Evaluation Metrics . . . 128
6.5 Evaluation . . . 128
6.5.1 Performance of Link Inference Attacks . . . 128
6.5.2 Ablation Studies . . . 132
6.6 Defense Strategies . . . 139
6.6.1 Lapgraph . . . 139
6.6.2 Label Perturbation . . . 141
6.6.3 Discussion . . . 143
6.7 Conclusion . . . 144

7 Conclusion and Future Work **147**
7.1 Conclusion . . . 147
7.2 Future Work . . . 149

List of Figures

2.1 Olivetti faces dataset and PCA components 17

2.2 Representation of a graph G with its corresponding adjacency matrix A and feature matrix X. Node colors indicate different classes. Dashed lines show how graph edges correspond to adjacency matrix entries. 22

2.3 Illustration of homophily in graphs. Left: Graph G with high homophily, where nodes of the same color (representing the same class) are more likely to be connected. Right: Graph G' with lower homophily, showing more connections between nodes of different colors. 23

2.4 Illustration of one iteration of the message passing process in GNNs, focusing on node F. (a) Initial Node Features: The original graph structure with each node's feature vector (x_1, x_2, x_3). (b) Message Reception: Node F receives messages $(m_{D\to F}, m_{G\to F})$ from its neighbors D and G. (c) Message Aggregation: F aggregates the received messages, combining information from its neighbors. (d) Node Update: F updates its features based on the aggregated information and its previous state. This process is repeated for multiple iterations, allowing nodes to integrate information from their expanding neighborhood, enabling GNNs to learn both local and global graph structures. The color intensity of nodes indicates their relevance in each step, with F being the focus throughout the process. 26

2.5 Illustration of a 2-layer GNN computation, focusing on node F. Left: The initial graph structure with node features. Right: The computational graph over two layers. In Layer-1, nodes D and G aggregate information from their neighbors (including F). In Layer-2, F's representation is computed from D and G's Layer-1 representations. Note that this simplified view does not show F's Layer-1 representation contributing to its Layer-2 representation, which would typically occur in standard GNN operations. Each AGG+UPDATE box represents both the aggregation of neighbor information and the update of the node's representation. This process allows F to capture information from its 2-hop neighborhood, illustrating how GNNs incorporate both local and increasingly global graph structure. 28
2.6 Comparison of transductive and inductive learning in GNNs. (a) Transductive learning: the model is trained on all nodes, including unlabeled ones. (b) Inductive learning: the model learns transferable functions that can generalize to unseen nodes not present during training. 29
2.7 Comparison of Horizontal and Vertical Federated Learning (Inspired by [84]) 34
2.8 Vertical Federated Graph Learning (VFGL) System Architecture 37
2.9 Illustration of edge-level differential privacy. The graphs G and G' are edge-level adjacent, differing only by the presence or absence of a single edge (highlighted in red). 46
2.10 Illustration of node-level differential privacy. The graphs G and G' are node-level adjacent, differing by the presence or absence of a single node and all its incident edges (highlighted in red). 47

4.1 Histogram of reconstruction errors for training and test samples in the Olivetti faces dataset using 50 principal components. The inset images show comparisons of original (left) and reconstructed (right) face samples for both low and high reconstruction errors. 70
4.2 Impact of the sample size N and the observed top-k components on the attack's performance. Shaded areas show 95% confidence intervals for the mean. 74
4.3 The AUC of the attack when the AG algorithm (a) and the Laplace vector query approach (b) are applied with various values of ε. Shaded areas are the 95% confidence intervals for the mean. 78

4.4 Attack performance with Laplacian approaches when the adversary intercepts all components ($k = d$). The infinity point represents the non-private case. 79

4.5 The hashed area shows where naive composition introduces less noise than advanced composition. 80

4.6 Trade-off posed by the four DP-PCA algorithms described in Section 4.3.1, between the total privacy budget ε and data utility. Utility is measured as the percentage of captured energy w.r.t. SVD. 81

4.7 Trade-off posed by the four DP-PCA algorithms described in Section 4.3.1, between attack performance and data utility. We measure attack performance through AUC, and utility through the percentage of captured energy w.r.t. SVD. 81

5.1 Adversary-Server Interaction: In the inference phase, the adversary first queries the prediction scores P of the target nodes, represented as $V_{\mathcal{A}}$. Next, the server sends the predictions P of the GNN to the adversary. Then, the adversary sends a *Connect* query to inject the malicious node v_m, with features x_m, to the target node v_t. Finally, after the injection, the adversary queries again the prediction scores P' of the target nodes $V_{\mathcal{A}}$. 90

5.2 Success rates of the NILS attack for different depths and malicious features generation strategies for the Twitch-FR dataset. 98

5.3 F_1 score of the NILS attack for different values of ε. 103

5.4 F_1 score utility of the GCN for different values of ε. 103

6.1 Schematic representation of a VFL setting with two clients and one server 111

6.2 Analysis of sample losses and cosine similarity threshold . . . 124

6.3 Overlap of predictions between gradient-based and label-based LIA in Photo dataset. 131

6.4 Evolution of the AUC for Gradient-based LIA (blue) and Intermediate-representation LIA (red) Over Time. The horizontal dashed lines indicate the maximum AUC achieved by the two attacks. Attacks were conducted at each training epoch, across five runs, with the mean and standard deviation of the AUCs reported. 133

6.5 Comparison of AUC between Gradient-based LIA (blue), Intermediate Representations-based LIA (red), and Feature-based LIA (orange) across different feature ratios of the adversary. The maximum AUC achieved by both attacks during training is reported, alongside the mean and standard deviation of these AUCs across five runs. 134
6.6 Distribution of the cosine similarities of gradients among linked pairs and unlinked pairs during different training epochs. . . . 135
6.7 Performance analysis with varying number of parties. The results show the AUC values across different datasets as the number of parties changes from 2 to 5. 136
6.8 Lapgraph defense results. $\varepsilon = \infty$ represents no defense; lower ε values indicate stronger privacy protection. 140
6.9 Label perturbation defense. $B = 0$ represents no defense; higher B values indicate stronger protection. 142

List of Tables

2.1 Table of Key Notations 22

4.1 Overview of the DP mechanisms aimed to protect PCA against MIA. Here, ε denotes the *total* privacy budget and ε' the *fraction* thereof assigned to each coefficient of A. 77

5.1 F_1 scores and standard deviations for different attack methods and datasets. 95
5.2 Comparative performance of NILS and LinkTeller across three datasets (TWITCH-FR, TWITCH-RU, and Flickr) under low, unconstrained, and high constraint settings. The results are presented in terms of precision and recall with corresponding standard deviations. 97
5.3 Comparative performance of NILS with LinkTeller [138] and link-stealing attacks in [54] across three datasets (Cora, Citeseer, and Pubmed). 98
5.4 Success rates of the attack for different depths in comparison with LinkTeller [138]. We use the all-ones strategy for NILS. . 99

6.1 Table of Notations 110
6.2 Adversary capabilities in different attack scenarios 116
6.3 Datasets statistics. 128
6.4 Accuracy of link inference attacks across datasets using GCN architecture. Bold numbers indicate highest accuracy among client-side attacks (first three columns). Participating clients can only conduct these three attacks. 130
6.5 Accuracy of gradient-based link inference attack on different GNN architectures 137
6.6 Accuracy of gradient-based LIA on ResNet-like architectures (Cora dataset) 138

Abstract

The increasing adoption of machine learning (ML) algorithms in privacy-sensitive domains has revolutionized data analysis across numerous fields. These algorithms, including Deep Neural Networks (DNN) as well as Principal Component Analysis (PCA) and Graph Neural Networks (GNNs), process vast amounts of data to extract valuable insights. The integration of these techniques into critical applications handling sensitive information, from healthcare records to social network data, has led to significant advances in data analysis and prediction capabilities. However, as these ML models become more prevalent, they inadvertently expose vulnerabilities that can compromise individual privacy through various attack vectors. This thesis addresses the critical privacy challenges posed by modern ML systems, focusing particularly on membership inference attacks (MIA) and link inference attacks (LIA). We propose novel attack methodologies and corresponding defensive measures that enhance our understanding of privacy vulnerabilities while providing practical solutions for privacy preservation. We aim to design these solutions while maintaining a careful balance between privacy protection and model utility, specifically for real-world ML applications. In our first contribution, we present a novel MIA against PCA, where an adversary can determine whether a specific data sample was used in computing the principal components. We demonstrate that this attack achieves high success rates, particularly when the number of samples used by PCA is relatively small. To counter this vulnerability, we investigate different approaches to implementing differential privacy mechanisms in PCA, analyzing their effectiveness in protecting against MIAs while preserving data utility. We provide comprehensive empirical evidence showing the trade-offs between privacy guarantees and model performance. Our second contribution is to introduce a new link inference attack, namely, the Node Injection Link Stealing (NILS) attack against GNNs that demonstrates how an adversary can exploit the dynamic nature of GNNs by injecting malicious nodes to infer edge information. We evaluate this attack based on a new differential privacy notion dedicated to graph structures and further propose dedicated defense strategies. Finally,

our third contribution focuses on the distributed setting, i.e., Vertical Federated Graph Learning (VFGL) and we develop a gradient-based LIA which reveals how gradient information in FL settings can leak graph structure details. Similarly to NILS, this attack is also evaluated across various datasets and model architectures and dedicated defensive strategies based on differential privacy mechanisms are proposed.

Résumé[1]

L'adoption croissante d'algorithmes d'apprentissage automatique dans des domaines sensibles à la protection de la vie privée a révolutionné l'analyse des données dans de nombreux domaines. Ces algorithmes, notamment les réseaux neuronaux profonds, l'analyse en composantes principales (PCA) et les réseaux neuronaux graphiques (GNN), traitent de grandes quantités de données pour en extraire des informations utilees et précieuses. L'intégration de ces techniques dans des applications critiques traitant des informations sensibles, comme des dossiers médicaux ou des données des réseaux sociaux, a permis des avancées significatives dans l'analyse des données et les capacités de prédiction. Cependant, à mesure que ces modèles de ML deviennent plus répandus, ils exposent involontairement des vulnérabilités qui peuvent compromettre la vie privée des individus par le biais de divers vecteurs d'attaque. Cette thèse aborde les défis critiques en matière de protection de la vie privée posés par les systèmes modernes de ML, en se concentrant particulièrement sur les attaques par inférence d'appartenance (MIA) et les attaques par inférence de lien (LIA). Nous proposons de nouvelles méthodologies d'attaque et des mesures défensives correspondantes qui améliorent notre compréhension des vulnérabilités en matière de protection de la vie privée tout en fournissant des solutions pratiques pour la protection de la vie privée. Nous visons à concevoir ces solutions tout en maintenant un équilibre prudent entre la protection de la vie privée et l'utilité du modèle, en particulier pour les applications de ML du monde réel. Dans notre première contribution, nous présentons une nouvelle attaque MIA contre le PCA, où un adversaire peut déterminer si un échantillon spécifique a été utilisé dans le calcul des composantes principales. Nous démontrons que cette attaque atteint des taux de réussite élevés, en particulier lorsque le nombre d'échantillons utilisés par le PCA est relativement faible. Pour contrer cette vulnérabilité, nous étudions différentes approches de mise en œuvre de mécanismes différentiels de protection de la vie privée (DP) dans le PCA, en analysant leur efficacité en matière de protection contre les MIAs tout en préservant l'utilité des données. Nous

[1]This summary was translated with the help of `deepl.com`

fournissons des preuves empiriques complètes montrant les compromis entre les garanties de confidentialité et la performance du modèle. Dans notre deuxième contribution, nous présentons une nouvelle attaque d'inférence de liens (LIA), à savoir l'attaque Node Injection Link Stealing (NILS) contre les GNN, qui démontre comment un adversaire peut exploiter la nature dynamique des GNNs en injectant des nœuds malveillants pour déduire des informations sur les arêtes. Nous évaluons cette attaque en fonction d'une nouvelle notion de confidentialité différentielle dédiée aux structures de graphes et proposons des stratégies de défense dédiées. Enfin, notre troisième contribution se concentre sur le cadre distribué, c'est-à-dire l'apprentissage vertical fédéré des graphes (VFGL), et nous développons une attaque LIA basée sur le gradient qui révèle comment les informations sur le gradient dans le FL peuvent laisser échapper des détails sur la structure des graphes. Comme pour NILS, cette attaque est également évaluée sur différents ensembles de données et architectures de modèles, et des stratégies défensives spécifiques basées sur des mécanismes DP sont proposées.

Chapter 1

Introduction

1.1 Motivation

The advancements in machine learning have empowered its use in diverse fields, transforming how data is analyzed, interpreted, and utilized. In sectors like healthcare [80, 36, 91], finance [55, 37, 46, 16], and social networks [47, 45, 102], machine learning models provide critical insights by processing vast datasets. These models, however, are often trained on sensitive information, raising significant concerns about data privacy [143]. Ideally, machine learning should recognize and generalize patterns across a population without compromising individual data privacy. However, in practice, models trained on confidential data sometimes reveal sensitive details about individuals within the training dataset [123], presenting privacy risks that go beyond their intended purpose.

Privacy in machine learning encompasses multiple aspects of data protection and presents unique challenges that go beyond traditional data security [99]. At the individual level, it involves protecting personal information, including demographic data, behavioral patterns, and sensitive attributes that could identify or characterize a person [2]. At the relationship level, it concerns protecting connections and interactions between entities, particularly in networked data structures [54]. An attacker's goals in this context can vary significantly: from identifying individuals in the training data, to inferring sensitive attributes [146], to uncovering hidden relationships between entities. These privacy requirements become increasingly complex as machine learning architectures evolve and capture more sophisticated patterns in data.

The evolution of machine learning architectures has introduced new and diverse privacy challenges [99]. While traditional neural networks primarily

faced risks related to individual data leakage, modern architectures present more complex privacy concerns. For instance, Graph Neural Networks (GNNs) [115] explicitly model relationships between entities, making both node features and edge connections potential sources of privacy leakage [54]. Similarly, federated learning systems, while designed to enhance privacy through distributed training [87], may inadvertently leak information through model updates and gradients [42]. Each of these architectures requires specific consideration of what constitutes private information and how it might be compromised.

Privacy attacks in machine learning can be classified based on their objectives and mechanisms. Membership Inference Attacks (MIA) [120] represent a fundamental privacy threat, with significant real-world implications. These attacks aim to determine if a particular individual's data was used during model training, which can have severe consequences in sensitive domains. For instance, identifying that a clinical record was used to train a disease-specific model could reveal an individual's medical condition. Such privacy breaches have drawn attention from regulatory bodies, with the National Institute of Standards and Technology (NIST) [125] explicitly classifying successful membership inference as a confidentiality violation. Furthermore, these privacy risks have broader implications under regulations like the General Data Protection Regulation (GDPR) [1], potentially classifying machine learning models as processors of personal information [129] and subjecting them to strict privacy requirements. In graph-structured data, this privacy concern extends to Link Inference Attacks (LIA) [54], which can be viewed as a form of membership inference for relationships, attempting to uncover connections between entities such as social network interactions or communication patterns. These attacks demonstrate how machine learning models, while designed to capture general patterns, can inadvertently leak specific information about their training data, posing risks for both individuals and organizations deploying machine learning services.

The growing recognition of these privacy risks has prompted the development of various protective measures [32, 99]. Traditional privacy-preserving approaches like encryption protect data during storage and transmission but fail to address inference attacks that operate on model outputs. Cryptographic techniques such as Multiparty Computation (MPC) [92, 145] and Homomorphic Encryption (HE) [43, 44] enable secure computation by allowing operations on encrypted data. However, while these methods effectively protect data during computation, they present significant limitations for deployed machine learning services. Many real-world applications require models to provide clear, unencrypted predictions to end-users for transparency, interpretability, and decision-making purposes [120, 94]. This fundamen-

tal requirement means that even with secure computation, the final model outputs remain vulnerable to inference attacks. Additionally, these cryptographic solutions introduce substantial computational overhead and require complex key management systems, making them impractical for many deployment scenarios [99]. In contrast, Differential Privacy (DP) [32, 30] offers a mechanism to protect against inference attacks by introducing controlled randomness into the learning process itself, potentially providing privacy guarantees even when model predictions must be exposed. The challenge lies in balancing this protection with model utility, as privacy guarantees often come at the cost of reduced accuracy or performance [32, 146, 99].

The motivation for this thesis is therefore twofold: to deepen the understanding of privacy vulnerabilities in machine learning systems, examining how and why attacks like MIA and LIA succeed, and to explore defense mechanisms that strengthen these systems against privacy risks. This work is particularly timely given the increasing regulatory requirements for data protection [113] and the growing deployment of machine learning in privacy-sensitive domains. Focusing on differential privacy as a defense strategy, this work aims to bridge the gap between model utility and privacy, contributing to the safe and ethical deployment of machine learning models in settings where data privacy is paramount.

1.2 Machine Learning Techniques and Privacy

In this thesis, we explore the intersection of machine learning and privacy, focusing on specific techniques—namely *Principal Component Analysis (PCA)*, *Graph Neural Networks (GNNs)*, and *Federated Learning*—that are widely used in privacy-sensitive domains such as healthcare [80, 36, 91], finance [55, 37, 46, 16], and social networks [47, 45, 102]. Understanding these core methods is essential to framing the privacy risks and vulnerabilities addressed in this work, as each technique presents unique challenges requiring careful evaluation and mitigation.

Principal Component Analysis (PCA) [57, 101] is a foundational unsupervised learning algorithm for dimensionality reduction, frequently applied in high-dimensional datasets to simplify analysis and improve efficiency. By transforming data into a lower-dimensional space, PCA enables easier handling of large datasets, retaining significant patterns, which is especially valuable in fields like bioinformatics and image processing. However, when applied to sensitive data, PCA's projections—its principal components—may retain individual-level information, creating privacy risks.

Graph Neural Networks (GNNs) [159, 140, 130] are a rapidly evolv-

ing class of models specifically designed to learn from graph-structured data, where entities and their relationships are encoded as nodes and edges. GNNs excel in applications such as social network analysis, recommendation systems, and fraud detection by capturing relational patterns within interconnected data. However, the networked structure of graph data can inherently contain sensitive information, particularly in social networks, where user relationships are often confidential.

Federated Learning [87, 144] is an emerging paradigm addressing privacy concerns by training models across decentralized devices, thus avoiding the need to centralize raw data. This approach is especially relevant in mobile applications and collaborative studies, where user privacy is a priority. However, even with data stored locally, federated learning systems can still be vulnerable to privacy attacks, as model updates or gradients shared between devices and servers may leak sensitive information, such as individual data contributions. This thesis examines privacy vulnerabilities in federated learning, with a focus on vertical federated graph learning scenarios, where inference attacks can target relationships between users or entities in graph-based federated settings.

Together, these techniques highlight the substantial privacy risks that arise when machine learning systems handle sensitive data. While enabling powerful analytical and predictive capabilities, these methods also expose vulnerabilities that adversaries can exploit to extract private information. The goal of this thesis is to analyze these risks comprehensively and to explore defenses, with a particular focus on *Differential Privacy (DP)* as a robust, adaptable mechanism to limit information leakage while preserving model utility [32, 30]. By understanding and mitigating these vulnerabilities, this research aims to contribute to safer and more ethical deployment of machine learning in privacy-sensitive contexts.

1.3 Privacy Attacks in Machine Learning

Privacy attacks in machine learning aim to exploit vulnerabilities in models to extract information about specific data points, often exposing sensitive details about individuals or relationships within the data. Among these, *Membership Inference Attacks (MIA)* and *Link Inference Attacks (LIA)* represent two key types of privacy threats, each posing unique challenges that this thesis will address.

Membership Inference Attacks (MIA) [120] are designed to determine whether a particular data sample was part of a model's training dataset. These attacks exploit subtle differences in model behavior or con-

fidence scores between data points seen during training and those that were not, using this information to infer the presence of specific samples in the training set. MIA are particularly concerning in applications where training data consists of private or sensitive information, such as in healthcare or finance, as they can inadvertently reveal individual-level information about the dataset [120]. In this thesis, we investigate MIA against Principal Component Analysis (PCA), a commonly used dimensionality reduction technique, examining both the nature of this vulnerability and the effectiveness of differential privacy as a defense.

Link Inference Attacks (LIA) [54] target graph-structured data, aiming to infer the existence or absence of edges between nodes in a graph. These edges often represent sensitive relationships, such as friendships in social networks or professional connections in organizational networks. LIA leverage the structure and outputs of Graph Neural Networks (GNNs) to deduce connections that may not be visible to a typical user, posing significant privacy risks, especially in social and professional networking applications [54]. This thesis examines LIA in the context of GNNs, exploring their mechanisms and proposing defense strategies to protect relational privacy within graph-structured data.

Together, MIA and LIA highlight the substantial privacy risks posed by machine learning models that process sensitive data. These attacks not only exploit model outputs and structures but also challenge the primary goal of machine learning: to recognize general patterns without exposing individual-level information. This thesis seeks to address these vulnerabilities by analyzing the effectiveness of privacy-preserving methods, with a particular emphasis on differential privacy, to safeguard against these types of privacy attacks.

1.4 Defenses Against Privacy Attacks in Machine Learning

To address the privacy risks inherent in machine learning, several privacy-preserving techniques have emerged. Some involve model regularization and noise addition [116], which aim to reduce overfitting and prevent models from memorizing specific data points. Techniques such as dropout [111], weight regularization [120, 93], and gradient clipping have been shown to limit a model's susceptibility to privacy attacks. However, these methods offer heuristic rather than formal guarantees and may still leave models vulnerable, particularly when adversaries employ sophisticated inference techniques.

Differential Privacy (DP) [32, 30] has emerged as a particularly effective and adaptable approach. Unlike other methods, DP provides formal privacy guarantees by introducing controlled noise into data or model outputs, making it mathematically provable that individual data points have minimal impact on the model's results. This characteristic aligns well with the primary goal of machine learning: to generalize patterns across data without revealing specific information about individual records.

In the context of this thesis, Differential Privacy is central due to its robustness in protecting against both MIA and LIA. For MIA, DP's noise ensures that model outputs do not reveal whether specific data points were included in the training set, thereby safeguarding individual privacy. For LIA, DP can reduce sensitivity to particular relationships within graph-structured data, helping to obscure connections in graph neural networks and preventing adversaries from deducing private relationships.

By focusing on Differential Privacy as the core defense mechanism, this thesis seeks to explore its potential to mitigate privacy vulnerabilities in machine learning systems and demonstrate how DP can effectively bridge the gap between privacy and model utility, supporting the safe deployment of machine learning applications in sensitive fields.

1.5 Problem Statement and Objectives

The increasing deployment of machine learning models in privacy-sensitive applications has underscored significant vulnerabilities related to data privacy. While machine learning aims to recognize general patterns across data, models can inadvertently expose specific information about individuals within the training set. This risk is particularly pronounced in applications involving sensitive data, where the privacy of personal and relational information is paramount. Consequently, this thesis focuses on understanding and mitigating the privacy vulnerabilities associated with machine learning, particularly with respect to dimensionality reduction techniques, graph-structured data, and federated learning.

The central problems addressed in this thesis are as follows:

- **Privacy Risks in Dimensionality Reduction:** Techniques such as Principal Component Analysis (PCA) are widely used for dimensionality reduction and data simplification, especially in high-dimensional datasets. However, PCA-transformed data can be susceptible to *Membership Inference Attacks (MIA)*, where adversaries infer whether specific data points were part of the original training set. This poses a critical privacy risk, especially when PCA is applied to sensitive datasets.

The thesis seeks to investigate these vulnerabilities, exploring how MIA operate in the context of PCA and assessing the extent to which differential privacy can mitigate these risks.

- **Privacy Challenges in Graph-Structured Data:** Graph Neural Networks (GNNs) have become a powerful tool for analyzing relational data in applications like social network analysis and recommendation systems. However, the unique structure of graph data introduces specific privacy risks, particularly through *Link Inference Attacks (LIA)*, which aim to reveal hidden connections between entities. This thesis examines the mechanisms by which LIA exploit GNNs and evaluates defense strategies, focusing on how differential privacy can safeguard relational information within graph data.

- **Privacy Vulnerabilities in Federated Learning:** Federated learning enables decentralized model training across multiple devices, aiming to preserve privacy by keeping raw data local. Yet, federated learning systems remain vulnerable to privacy leaks through shared gradients or model updates, potentially exposing individual contributions in collaborative settings. In this thesis, we investigate these vulnerabilities within the context of vertical federated graph learning, where LIA can target sensitive connections within distributed graph-structured data. The research aims to assess the effectiveness of differential privacy in mitigating these risks in federated learning environments.

Based on these challenges, the primary objectives of this thesis are:

- **Objective 1:** To construct a membership inference attacks in PCA and evaluate the impact of differential privacy as a defense mechanism.

- **Objective 2:** To investigate link inference risks in graph neural networks, identifying key vulnerabilities and proposing differential privacy mechanisms to protect sensitive relationships.

- **Objective 3:** To assess the privacy risks associated with federated learning in graph-based settings, focusing on the potential for link inference attacks and evaluating the feasibility of differential privacy as a defense mechanism.

By addressing these objectives, this thesis aims to contribute to the field of privacy-preserving machine learning by providing a comprehensive understanding of specific privacy attacks and defenses. Ultimately, this research seeks to advance the safe deployment of machine learning models in applications where data privacy is essential.

1.6 Contributions and Outline

This thesis addresses critical privacy challenges in machine learning, focusing on dimensionality reduction, graph-structured data, and federated learning. Through an analysis of privacy risks and an exploration of defense mechanisms, particularly differential privacy, this research contributes to advancing the privacy-preserving deployment of machine learning models in sensitive applications.

Chapter 2 lays the foundation for the thesis by providing a comprehensive background on core machine learning techniques, privacy attacks, and defense mechanisms. It introduces Principal Component Analysis (PCA) for dimensionality reduction, Graph Neural Networks (GNNs) for handling graph-structured data, and Differential Privacy (DP) as a promising approach for protecting individual data privacy. This chapter also covers a variety of privacy attack types, offering essential context for the analyses and solutions developed in subsequent chapters.

Following this, Chapter 3 presents a review of related work in privacy-preserving machine learning, with a particular focus on membership inference attacks, link inference attacks, and privacy risks in federated learning environments. This review identifies gaps in existing research, highlighting specific areas where privacy vulnerabilities persist and where further investigation is needed.

Building on this foundation, Chapter 4 explores the susceptibility of PCA to membership inference attacks. This chapter details the attack methodology and provides an experimental evaluation, assessing the effectiveness of differential privacy in mitigating membership inference risks within PCA-transformed data. By analyzing the privacy-utility trade-offs in PCA, this chapter provides insights into safeguarding dimensionality reduction techniques.

In Chapter 5, the thesis introduces the Node Injection Link Stealing (NILS) attack, a novel approach to link inference attacks that targets GNNs. This chapter explains the design and implementation of NILS, compares its effectiveness to existing link inference methods, and evaluates the potential of differential privacy-based defenses to protect relational privacy within graph-based models.

Chapter 6 addresses privacy vulnerabilities in federated learning, specifically within the context of vertical federated graph learning. Here, we develop a gradient-based link inference attack that demonstrates how shared gradients can leak sensitive connection information in distributed settings. This chapter also examines defense mechanisms, including differential privacy, assessing their effectiveness in reducing the privacy risks associated

with collaborative learning.

Finally, Chapter 7 concludes the thesis with a summary of key contributions and a discussion of future research directions. It synthesizes the findings and reflects on the implications of this work for the field of privacy-preserving machine learning, identifying potential areas for further exploration and development.

1.7 List of Publications

The research presented in this thesis has led to the following publications:

- Oualid Zari, Javier Parra-Arnau, Ayşe Ünsal, Thorsten Strufe, and Melek Önen. "Membership Inference Attack Against Principal Component Analysis," in *Proceedings of the International Conference on Privacy in Statistical Databases (PSD)*, 2022.
- Oualid Zari, Javier Parra-Arnau, Ayşe Ünsal, and Melek Önen. "Node Injection Link Stealing Attack Against Graph Neural Networks," in *Proceedings of the International Conference on Privacy in Statistical Databases (PSD)*, 2024.
- Oualid Zari, Chuan Xu, Javier Parra-Arnau, Ayşe Ünsal, and Melek Önen. "Link Inference Attacks in Vertical Federated Graph Learning," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2024.

Chapter 2

Background

The rapid advancement of machine learning has revolutionized numerous fields, from social networks to healthcare, by enabling the analysis and interpretation of vast amounts of data. However, this progress has also raised significant privacy concerns, particularly when machine learning techniques are applied to sensitive data. In this chapter, we thoroughly explore the machine learning methods, privacy attacks, and defense mechanisms that are central to our research on privacy in machine learning.

We begin by introducing fundamental machine learning techniques relevant to our work, focusing on Principal Component Analysis (PCA) [57, 101] and Graph Neural Networks (GNNs) [159, 140, 130]. Additionally, we explore Federated Learning, a distributed machine learning approach that aims to train models on decentralized data while maintaining data isolation. PCA, a widely used unsupervised learning method, serves as a crucial tool for dimensionality reduction and data analysis. GNNs, on the other hand, represent a powerful class of models designed to handle graph-structured data, which are increasingly prevalent in modern applications.

Following this, we delve into the critical intersection of privacy and machine learning. We explore various types of privacy attacks that have emerged in recent years, with a particular focus on Membership Inference Attacks (MIA) and Link Inference Attacks (LIA). These attacks highlight the potential vulnerabilities in machine learning models and underscore the importance of privacy-preserving techniques.

To address these privacy concerns, we then introduce key privacy-preserving methodologies. We provide an in-depth discussion of Differential Privacy (DP), a rigorous mathematical framework for quantifying and bounding privacy leakage.

By covering these topics, this chapter aims to provide a solid foundation for understanding the complex interplay between machine learning tech-

niques and privacy considerations. This background will be essential for understanding the novel contributions presented in the subsequent chapters of this thesis, where we explore advanced privacy attacks and propose new defense strategies in the context of PCA and GNNs.

2.1 Foundations of Machine Learning

This section introduces two fundamental machine learning techniques that are central to our research at the intersection of machine learning and privacy: Principal Component Analysis (PCA) and Graph Neural Networks (GNNs). We provide a comprehensive overview of each technique, including their mathematical foundations, implementations, and applications.

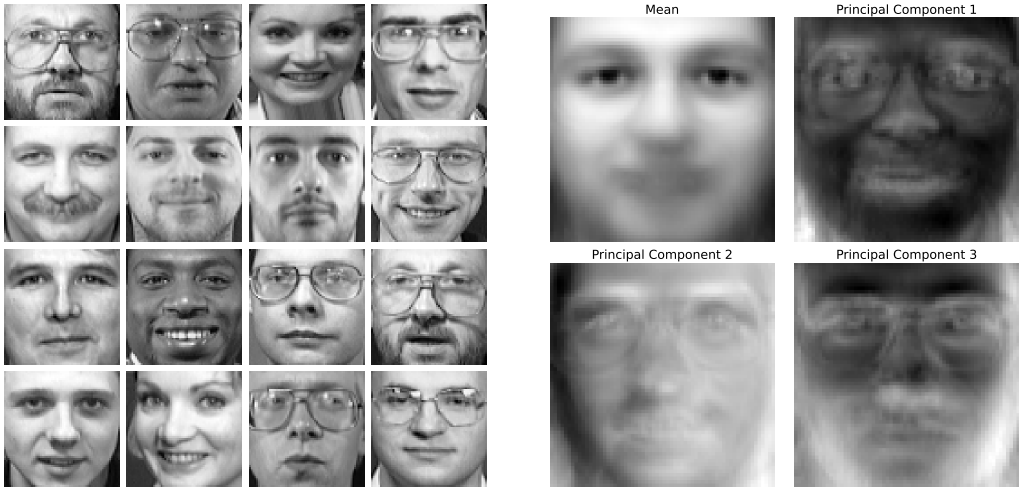
2.1.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is one of the most widely used unsupervised machine learning algorithms for dimensionality reduction and data analysis. Over the past decade, PCA has found application in a vast and rapidly growing number of systems for analyzing and classifying often privacy-sensitive data [158, 128, 122].

PCA is primarily employed for summarizing the information content in databases by reducing the dimensionality of the data while preserving as much variability as possible. The output of this statistical tool is a set of *principal components* whose size is usually much smaller than the total number of attributes of the underlying data. These principal components are orthogonal vectors that capture the directions of maximum variance in the data.

Example of PCA Application: Face Recognition

To illustrate the concept of PCA in a real-world scenario, we will examine its application to face recognition using the Olivetti faces dataset [112]. This dataset consists of 400 grayscale images of faces, each 64x64 pixels, featuring 40 distinct subjects with 10 images per subject. It has been widely used in computer vision research to evaluate face recognition algorithms [128, 4]. Figure 2.1 provides an overview of the dataset and the results of applying PCA. Figure 2.1a displays a random selection of 16 face images from the dataset, illustrating the variety of faces, poses, and lighting conditions. Figure 2.1b shows the mean face and the first three principal components (eigenfaces) resulting from PCA [128]. The mean face represents the average of all faces



(a) Random selection of face images (b) Mean face and principal components

Figure 2.1: Olivetti faces dataset and PCA components

in the dataset. The principal components capture the main modes of variation in the data [71]. The first component often relates to the overall lighting conditions, while subsequent components might capture variations in facial features or expressions [121]. By projecting the original 4096-dimensional face images (64x64 pixels) onto these principal components, we can represent each face as a vector of weights in a lower-dimensional space [128]. If we use sufficient dimensions (but fewer than the original 4096), we can use this lower-dimensional representation as input to various machine learning algorithms, such as nearest-neighbor classifiers [20] or support vector machines [19], to perform face recognition [13, 103]. This dimensionality reduction approach offered by PCA is often faster and more reliable than working directly in pixel space for face recognition tasks [13, 157].

Mathematical Formulation of PCA

Given a set $D = \{x_n \in \mathbb{R}^d : n = 1 : N\}$ of N raw data samples corresponding to N individuals of dimension d, we denote the data matrix where each column is a data sample by $X = [x_1, \ldots, x_N]$. We assume that data X has zero mean, which can be ensured by centering the data.

The goal of PCA is to find a linear transformation that maps the original d-dimensional data to a lower k-dimensional space (where $k < d$) while maximizing the variance of the projected data. Mathematically, this can be formulated as an optimization problem:

$$\max_{W} \operatorname{tr}(W^T X X^T W) \tag{2.1}$$

$$\text{subject to } W^T W = I \tag{2.2}$$

where $W \in \mathbb{R}^{d \times k}$ is the transformation matrix, and I is the $k \times k$ identity matrix. The constraint ensures that the columns of W are orthonormal.

This optimization problem can be solved using the eigen-decomposition of the covariance matrix $A = \frac{1}{N} X X^T$. The solution W consists of the top k eigenvectors of A, corresponding to the k largest eigenvalues. These eigenvectors are the principal components.

Formally, if we decompose A as $A = V \Lambda V^T$, where $V = [v_1, ..., v_d]$ are the eigenvectors and $\Lambda = \text{diag}(\lambda_1, ..., \lambda_d)$ are the corresponding eigenvalues (sorted in descending order), then $W = [v_1, ..., v_k]$.

This formulation reduces the dimension because it projects the original d-dimensional data onto a k-dimensional. subspace spanned by the top k eigenvectors, which capture the directions of maximum variance in the data.

Reconstruction Error and PCA Optimization

While the primary goal of PCA is dimension reduction, it can also be formulated as a minimization problem of the reconstruction error. This perspective provides insights into how well PCA preserves information.

For a given data point x_n, its reconstruction $\hat{x}_n$ after PCA is:

$$\hat{x}_n = W W^T x_n \tag{2.3}$$

The reconstruction error for this point is:

$$\mathcal{L}_n = \|x_n - \hat{x}_n\|_2^2 = \|x_n - W W^T x_n\|_2^2 \tag{2.4}$$

The average reconstruction error over all the data points is:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_n = \frac{1}{N} \sum_{n=1}^{N} \|x_n - W W^T x_n\|_2^2 \tag{2.5}$$

Minimizing this average reconstruction error is equivalent to the maximization problem in Equation 2.1. The matrix W that minimizes $\mathcal{L}$ is the same as the one that maximizes the variance of the projected data.

Measuring the Utility of PCA

To evaluate the effectiveness of Principal Component Analysis (PCA) in dimensionality reduction, we measure utility based on the proportion of the total variance (also known as energy) captured by the selected principal components. Specifically, we are interested in how much of the original data's variance is retained when projecting onto a lower-dimensional subspace spanned by the top k principal components.

Given the zero-mean data matrix $X \in \mathbb{R}^{d \times N}$, the sample covariance matrix is defined as:

$$A = \frac{1}{N} X X^T \tag{2.6}$$

Let $V = [v_1, v_2, \ldots, v_d]$ be the matrix of eigenvectors of A, with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$. The eigenvalues are sorted in descending order to prioritize the components capturing the most variance.

We define the **cumulative variance** captured by the top k principal components as:

$$\Gamma_k = \sum_{i=1}^{k} \lambda_i = \operatorname{tr}(V_k^T A V_k) \tag{2.7}$$

where $V_k = [v_1, v_2, \ldots, v_k] \in \mathbb{R}^{d \times k}$ contains the top k eigenvectors.

The utility q of using these k principal components is then defined as the fraction of the total variance they capture:

$$q = \frac{\Gamma_k}{\operatorname{tr}(A)} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \tag{2.8}$$

This measure q indicates the effectiveness of the dimensionality reduction: a higher value of q means that more of the original data's variance is retained in the reduced-dimensional representation.

In practical applications, we often select the reduced dimension k such that a predetermined threshold of the total variance is captured. For instance, choosing k so that $q \geq 90\%$ ensures that the lower-dimensional subspace retains at least 90% of the original data's variance.

When comparing different PCA methods or algorithms, such as standard PCA and alternative approaches, we can use this utility measure to assess their performance. Let $\hat{V}_k$ denote the principal components obtained from an alternative PCA method, and let $\hat{\lambda}_i$ be the corresponding eigenvalues. The cumulative variance captured by the alternative method is:

$$\hat{\Gamma}_k = \sum_{i=1}^{k} \hat{\lambda}_i = \mathrm{tr}(\hat{V}_k^T A \hat{V}_k) \tag{2.9}$$

The utility of the alternative method is then:

$$\hat{q} = \frac{\hat{\Gamma}_k}{\mathrm{tr}(A)} = \frac{\sum_{i=1}^{k} \hat{\lambda}_i}{\sum_{i=1}^{d} \lambda_i} \tag{2.10}$$

By comparing q and $\hat{q}$, we can quantify how well the alternative PCA method preserves the data's variance relative to the standard PCA. A higher value of $\hat{q}$ indicates that the method retains more of the original variance, thus offering better utility in terms of information preservation.

This utility measurement is crucial for understanding the trade-offs involved in dimensionality reduction. It provides a quantitative basis for selecting the number of principal components and for comparing different PCA methodologies based on how effectively they capture the essential structure of the data.

2.1.2 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) have emerged as a powerful class of machine learning models specifically designed to handle graph-structured data. These models have gained considerable attention in recent years due to their ability to effectively learn and capture complex patterns in graph data, showing remarkable performance across a wide range of tasks [115].

GNNs have found applications in numerous domains where data can be represented as graphs. Some key applications include:

- **Node Classification**: Predicting labels or attributes of nodes in a graph. For example, in social networks, this could involve predicting user interests or demographics based on their connections and profile information [131, 47].

- **Link Prediction**: Forecasting the likelihood of a link forming between two nodes. This is particularly useful in recommender systems, where the task might be to predict potential friendships or product recommendations [155].

- **Graph Classification**: Assigning labels to entire graphs. This has applications in areas such as molecule property prediction in chemistry, where each molecule is represented as a graph [142, 135].

- **Community Detection**: Identifying clusters or communities within a graph. This is valuable in social network analysis for understanding group dynamics and in biology for protein complex detection [118].

- **Graph Generation**: Creating new graphs that share similar properties with a given set of graphs. This has applications in drug discovery, where new molecular structures can be generated [59].

- **Traffic Prediction**: Modeling road networks as graphs to forecast traffic patterns and congestion [141].

- **Fraud Detection**: Identifying fraudulent activities in financial transaction networks by analyzing patterns of connections and behaviors [83].

While GNNs demonstrate versatility across these various tasks, this thesis primarily focuses on their application to node classification. In node classification, the objective is to assign labels to individual nodes based on their features and the overall graph structure. This task is particularly relevant in many real-world scenarios, such as predicting user attributes in social networks or classifying protein functions in biological networks [133, 80].

The choice to concentrate on node classification provides a specific context for our exploration of GNN architectures and their vulnerabilities. In the following sections, we will explain how GNNs work through the lens of node classification tasks, which will serve as a foundation for our later discussions on privacy concerns and link stealing attacks in Chapters 5 and 6.

Graph Data Structures and Properties

To lay the groundwork for our discussion of GNNs and their vulnerabilities, we first introduce the fundamental graph data structures and properties that underpin these models.

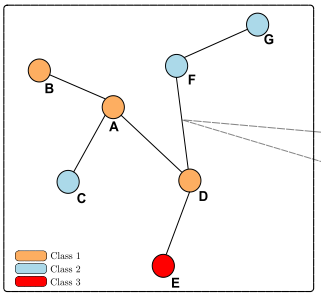
Notations Table 2.1 summarizes the key notations used consistently throughout this thesis.

Basic Definitions A graph $G = (V, E)$ consists of a set of nodes V and a set of edges E, as defined in Table 2.1. Nodes represent entities or objects in the data, while edges represent relationships or interactions between the nodes. For instance, in a social network graph, nodes might represent users, and edges could indicate friendships or interactions between users.

Table 2.1: Table of Key Notations

Notation	Description
G	Graph
V	Set of nodes in the graph
E	Set of edges in the graph
A	Adjacency matrix
X	Feature matrix
n	Number of nodes
d	Number of features
$\mathcal{Y}$	Set of node labels

Graph (G)



Adjacency Matrix (A)

	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	0	0	0	0	0
C	1	0	0	0	0	0	0
D	1	0	0	0	1	1	0
E	0	0	0	1	0	0	0
F	0	0	0	1	0	0	1
G	0	0	0	0	0	1	0

Feature Matrix (X)

	x_1	x_2	x_3
A	0.1	0.8	0.3
B	0.2	0.7	0.4
C	0.9	0.2	0.5
D	0.3	0.6	0.8
E	0.5	0.4	0.2
F	0.7	0.1	0.9
G	0.4	0.5	0.6

Figure 2.2: Representation of a graph G with its corresponding adjacency matrix A and feature matrix X. Node colors indicate different classes. Dashed lines show how graph edges correspond to adjacency matrix entries.

Figure 2.2 illustrates a simple graph along with its corresponding adjacency matrix and feature matrix. The adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $n = |V|$ is the number of nodes in the graph, represents the structure of the graph. In its simplest form, $A_{ij} = 1$ if there exists an edge between nodes i and j, and $A_{ij} = 0$ otherwise. This binary representation can be extended to weighted graphs, where A_{ij} can take on real values representing the strength or importance of the connection between nodes.

Nodes in a graph often have associated features or attributes, represented by a feature matrix $X \in \mathbb{R}^{n \times d}$, where d is the dimensionality of the feature space, as noted in Table 2.1. Each row in X corresponds to a node in the graph and contains that node's feature vector. In Figure 2.2, the feature matrix contains hypothetical features x_1, x_2, and x_3 for each node.

For supervised learning tasks such as node classification, a set of labels $\mathcal{Y}$ is associated with some or all of the nodes in the graph. These labels represent the target classes or categories that the model aims to predict for

unlabeled nodes. In our example, nodes are colored to represent different classes: orange for class 1, blue for class 2, and red for class 3.

The combination of the graph structure (represented by the adjacency matrix A) and node features (represented by the feature matrix X) forms the foundational data structure for GNNs.

Graph Properties Two properties of graphs that significantly influence the behavior and performance of GNNs are homophily and density. These properties play a crucial role in how information propagates through the graph and, consequently, how GNNs learn and make predictions.

Homophily Homophily refers to the tendency of similar nodes to connect with each other [88]. In the context of node classification, homophily implies that nodes with the same label are more likely to be connected than nodes with different labels.

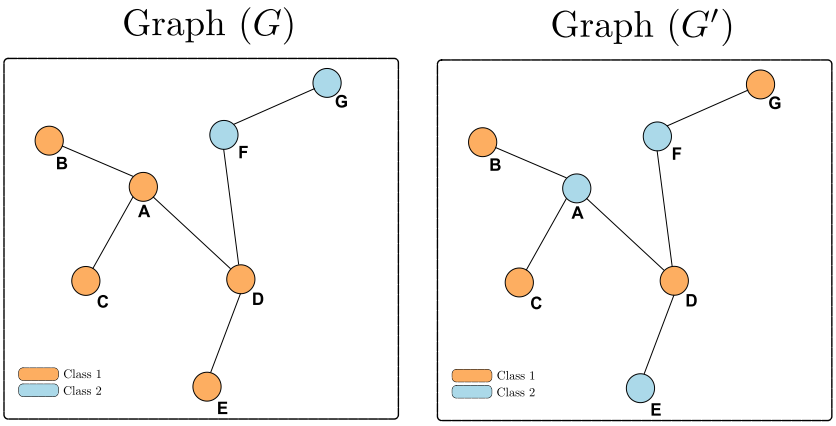


Figure 2.3: Illustration of homophily in graphs. Left: Graph G with high homophily, where nodes of the same color (representing the same class) are more likely to be connected. Right: Graph G' with lower homophily, showing more connections between nodes of different colors.

Figure 2.3 illustrates the concept of homophily in graphs. In the high homophily graph G (left), nodes of the same color (representing the same class) are more likely to be connected. In contrast, the lower homophily graph G' (right) shows more connections between nodes of different colors.

The degree of homophily in a graph can be quantified using various measures. One common approach is to calculate the edge homophily ratio:

$$h = \frac{\text{number of edges connecting same-class nodes}}{\text{total number of edges}} \tag{2.11}$$

A high value of h indicates stronger homophily in the graph.

Homophily is particularly relevant for GNNs because these models often rely on the assumption that connected nodes share similar characteristics or labels. This property influences how GNNs aggregate and propagate information across the graph during the learning process. In highly homophilous graphs, GNNs can more effectively leverage neighborhood information to make accurate predictions, potentially leading to better performance in node classification tasks [161].

As we will show in Chapter 6, homophily has a high impact on the success of link inference attacks. In graphs with strong homophily, attackers can exploit the tendency of similar nodes to connect, making it easier to infer the existence of edges between nodes with similar labels or features. This relationship between homophily and attack vulnerability underscores the importance of understanding and quantifying this property in the context of graph privacy.

Density Graph density measures how many edges are present in a graph compared to the maximum possible number of edges. For an undirected graph, density is calculated as:

$$\text{Density} = \frac{2|E|}{n(n-1)} \tag{2.12}$$

where $|E|$ is the number of edges and n is the number of nodes. The density value ranges from 0 (a graph with no edges) to 1 (a complete graph where every node is connected to every other node).

Graph density affects the performance of GNNs in several ways:

- **Information Flow**: In denser graphs, information can propagate more easily between nodes, potentially leading to more effective learning in GNNs [147].
- **Overfitting**: However, very high density can sometimes lead to oversmoothing in GNNs, where node representations become too similar, making it harder to distinguish between different classes [142].
- **Computational Complexity**: The density of a graph also affects the computational requirements of GNNs, with denser graphs generally requiring more computational resources [147].

In the context of link stealing attacks, graph density can influence the attacker's ability to infer edges. In sparse graphs, the presence of an edge might

reveal more information than in dense graphs, potentially making sparse graphs more vulnerable to certain types of attacks.

Understanding these graph properties is crucial for analyzing the behavior of GNNs in node classification tasks and for assessing their vulnerability to link stealing attacks. In the following chapters, we will explore how these properties interact with various GNN architectures and how they can be exploited or protected against in the context of link privacy.

Core Principles of GNNs

GNNs have revolutionized the field of graph-based machine learning by effectively capturing and leveraging the structural information inherent in graph data. At the heart of GNNs lie two fundamental principles: the message passing framework and the learning paradigms. These principles form the foundation upon which various GNN architectures are built and enable these models to effectively tackle a wide range of graph-based tasks.

Message Passing Framework The message passing framework is the cornerstone of GNNs, enabling them to process and learn from graph-structured data. This framework operates on the principle that each node in a graph can be represented by iteratively aggregating information from its local neighborhood. The process typically involves three main steps that are repeated for a fixed number of iterations or until convergence: *message reception*, *message aggregation*, and *node update*.

Figure 2.4 illustrates one iteration of the message passing process. In the initial state, each node has its own feature vector (panel a). During each iteration, the following steps occur:

1. **Message Reception**: A node receives messages from its neighboring nodes (panel b).

2. **Message Aggregation**: The node combines these received messages (panel c).

3. **Node Update**: The node updates its features using the aggregated information and its previous state (panel d).

This process is repeated for multiple iterations, allowing information to propagate through the graph and nodes to capture increasingly global context.

Mathematically, we can formulate the message passing framework for iteration k as follows:

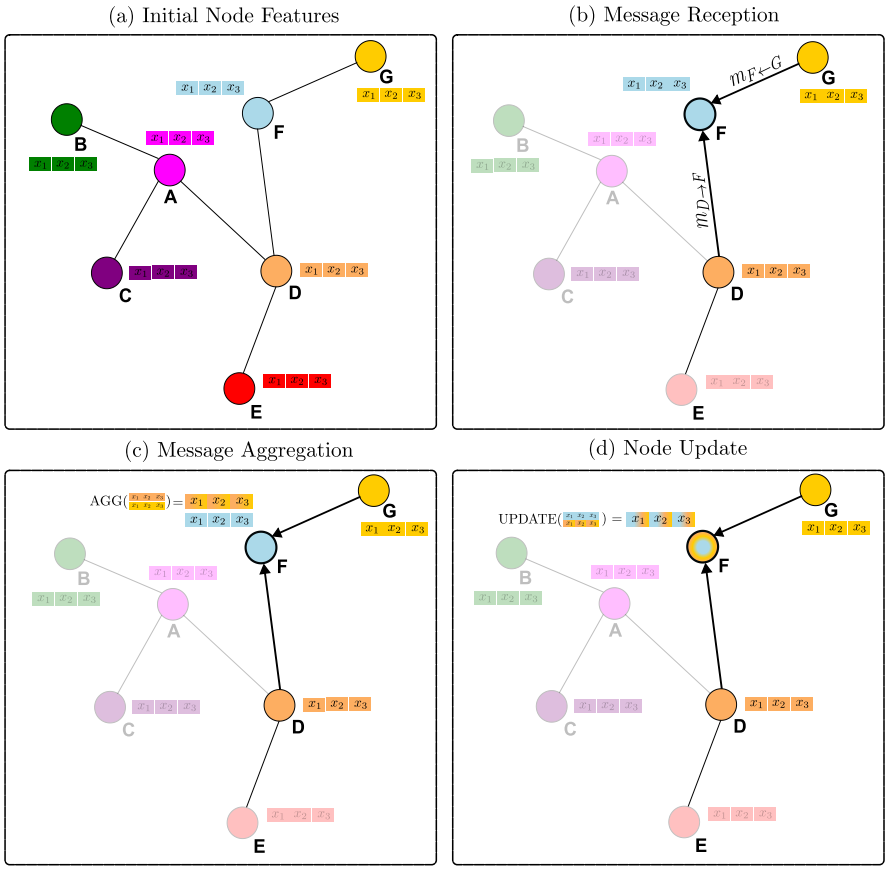


Figure 2.4: Illustration of one iteration of the message passing process in GNNs, focusing on node F. (a) Initial Node Features: The original graph structure with each node's feature vector (x_1, x_2, x_3). (b) Message Reception: Node F receives messages $(m_{D\to F}, m_{G\to F})$ from its neighbors D and G. (c) Message Aggregation: F aggregates the received messages, combining information from its neighbors. (d) Node Update: F updates its features based on the aggregated information and its previous state. This process is repeated for multiple iterations, allowing nodes to integrate information from their expanding neighborhood, enabling GNNs to learn both local and global graph structures. The color intensity of nodes indicates their relevance in each step, with F being the focus throughout the process.

$$h_v^{(k+1)} = \text{UPDATE}^{(k)}\left(h_v^{(k)}, \text{AGGREGATE}^{(k)}\left(\{m_{u\to v}^{(k)} : u \in \mathcal{N}(v)\}\right)\right) \quad (2.13)$$

where $h_v^{(k)}$ is the feature vector of node v at iteration k, $m_{u\to v}^{(k)}$ is the message from node u to node v at iteration k, $\mathcal{N}(v)$ is the neighborhood of node v, and AGGREGATE and UPDATE are learnable functions. The initial state $h_v^{(0)}$ is typically set to the input features of node v. After K iterations, the final node representations $h_v^{(K)}$ can be used for downstream tasks such as node classification or link prediction.

An alternative and insightful way to visualize the message passing mechanism in GNNs is through the lens of computational graphs. Computational graphs provide a structured representation of how information flows and is processed through the network over multiple iterations. This perspective is particularly useful for understanding the expanding receptive field of each node and how increasingly global information is incorporated into node representations.

To illustrate this concept, let us consider a computational graph for a depth-2 GNN, focusing on node F from our previous examples.

Figure 2.5 presents the computational graph for node F in a depth-2 GNN. This representation helps visualize how information flows through the network over multiple iterations, explicitly showing the expanding receptive field of node F. Let us break down this process in relation to our mathematical formulation:

1. Initial State (Layer 0): We begin with the original features of each node. For node F, this is represented as $h_F^{(0)}$.

2. First Iteration (Layer 1):

   - F's neighbors D and G aggregate information from their respective neighborhoods, including F.
   - This aggregation is represented by the AGG+UPDATE boxes in Layer 1.
   - Mathematically, for node D, this can be expressed using our general equation:
     $h_D^{(1)} = \text{UPDATE}^{(0)}(h_D^{(0)}, \text{AGGREGATE}^{(0)}(\{m_{u\to D}^{(0)} : u \in \mathcal{N}(D)\}))$
   - The same process occurs for node G.

3. Second Iteration (Layer 2):

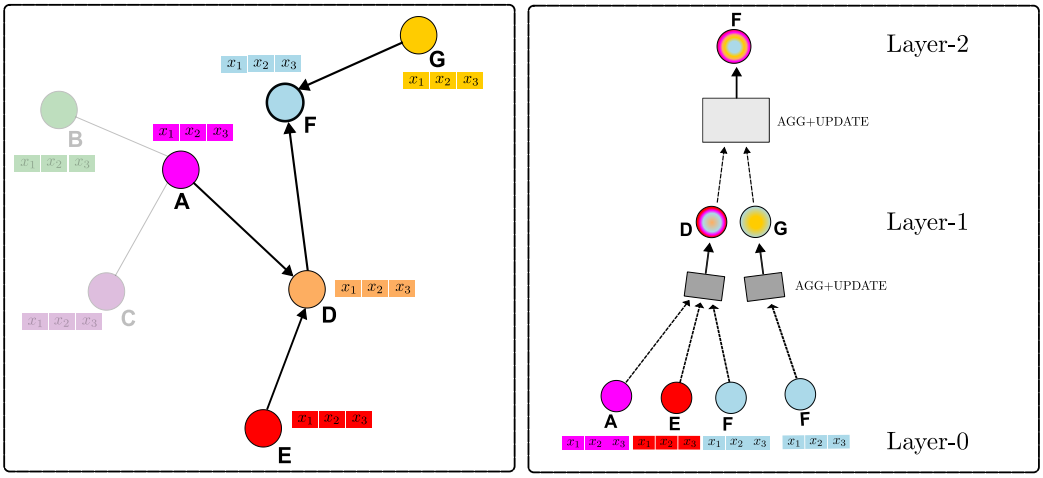


Figure 2.5: Illustration of a 2-layer GNN computation, focusing on node F. Left: The initial graph structure with node features. Right: The computational graph over two layers. In Layer-1, nodes D and G aggregate information from their neighbors (including F). In Layer-2, F's representation is computed from D and G's Layer-1 representations. Note that this simplified view does not show F's Layer-1 representation contributing to its Layer-2 representation, which would typically occur in standard GNN operations. Each AGG+UPDATE box represents both the aggregation of neighbor information and the update of the node's representation. This process allows F to capture information from its 2-hop neighborhood, illustrating how GNNs incorporate both local and increasingly global graph structure.

- F aggregates information from the updated representations of D and G.
- This is represented by the single AGG+UPDATE box in Layer 2.
- Mathematically, this is expressed as:
  $h_F^{(2)} = \text{UPDATE}^{(1)}(h_F^{(1)}, \text{AGGREGATE}^{(1)}(\{m_{D\to F}^{(1)}, m_{G\to F}^{(1)}\}))$

It is important to note that in standard GNN operations, F's Layer-1 representation would typically contribute to its Layer-2 representation. This self-dependency across layers, while not shown in the simplified figure, is crucial for maintaining and updating a node's own information throughout the network's depth.

This computational graph perspective illustrates how, after two iterations, F's final representation $h_F^{(2)}$ has incorporated information from its 2-hop neighborhood, including nodes A, B, C, and E. This expanding receptive

field is key to GNNs' ability to capture both local and increasingly global graph structures.

Understanding this iterative process and the expanding receptive field is crucial for our analysis of privacy threats. The way information propagates through the graph over multiple iterations plays a key role in how GNNs learn and represent node features. In Chapter 5, we will explore how this message passing mechanism can be exploited to infer the edges of the graph.

Learning Paradigms in GNNs GNNs support two main learning paradigms: *transductive learning* and *inductive learning*. These paradigms differ in how they handle unseen nodes and determine the scope of the learning process. Understanding these paradigms is crucial for grasping how GNNs learn and generalize, which in turn informs our analysis of their vulnerabilities to link stealing attacks.

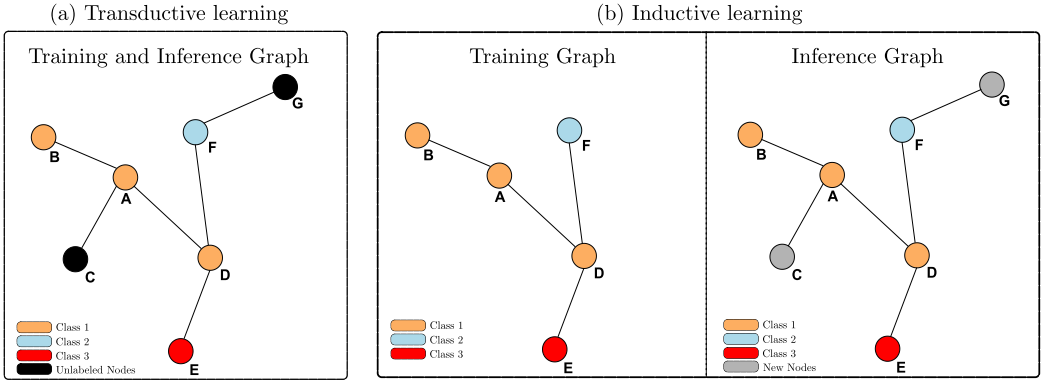


Figure 2.6: Comparison of transductive and inductive learning in GNNs. (a) Transductive learning: the model is trained on all nodes, including unlabeled ones. (b) Inductive learning: the model learns transferable functions that can generalize to unseen nodes not present during training.

Figure 2.6 illustrates the difference between transductive and inductive learning in GNNs. Transductive learning assumes that the entire graph structure is known during training, including unlabeled nodes. As shown in Figure 2.6(a), all nodes (A-G) are present in a single "Training and Inference Graph". This approach allows the model to leverage information from all nodes, even those without labels, during the learning process. However, transductive models are limited to making predictions only on nodes that were present during training. In contrast, inductive learning enables GNNs to generalize to entirely new, unseen nodes or even new graphs. Figure 2.6(b) demonstrates this by showing separate "Training Graph" and "Inference Graph". The inference graph includes new nodes (G and C, shown

in grey) that were not present during training. This capability is achieved by learning a set of aggregation functions that can be applied to any node, regardless of whether it was present in the training data.

Recall from Section 2.1.2 that GNNs operate through a message passing mechanism, where nodes iteratively aggregate information from their neighbors. In both transductive and inductive settings, what the model actually learns during training are the parameters of the neural networks that perform this aggregation and update process.

For instance, in a two-layer GNN, the model learns:

- Parameters for the first layer's aggregation function
- Parameters for the first layer's update function
- Parameters for the second layer's aggregation function
- Parameters for the second layer's update function

These learned parameters correspond to the "AGG+UPDATE" boxes shown in Figure 2.5. The key difference is how these learned functions are applied:

In transductive learning, these functions are applied to a fixed set of nodes, allowing the model to capture node-specific information. In inductive learning, these functions are designed to be applicable to any node in any graph, based solely on the node's features and local neighborhood structure.

This inductive capability allows GNNs to generalize not just to new nodes within the same graph, but to entirely new graphs. The learned aggregation and update functions can be applied to any graph structure, as long as the node features are compatible with the input dimensions the model was trained on.

The inductive nature of GNNs is particularly relevant to our research on link stealing attacks, especially in the context of node injection attacks. As we will explore in Chapter 5, attackers can exploit this by injecting new nodes into the graph and observing how the GNN processes these additions. This ability to handle unseen nodes opens up new avenues for potential privacy breaches, highlighting the importance of developing robust privacy-preserving techniques for GNNs.

In the following, we will delve into specific GNN architectures, starting with Graph Convolutional Networks. We will examine how these architectures implement the message passing mechanism and how their design choices affect their learning capabilities.

GNN Architectures

Having explored the core principles and learning paradigms of GNNs, we now examine specific GNN architectures. These architectures implement the message passing mechanism in various ways, each with its own strengths and potential vulnerabilities. We focus on three prominent GNN architectures: Graph Convolutional Networks (GCN), GraphSAGE, and Graph Attention Networks (GAT).

Graph Convolutional Networks (GCN) GCNs [75] adapt the concept of convolution to graph-structured data. The core idea is to generate node embeddings by aggregating information from a node's local neighborhood. For a graph $G = (V, E)$, the GCN layer for a node v is formulated as:

$$h_v^{(l+1)} = \sigma(\underbrace{W^{(l)}}_{\text{Update}} \underbrace{\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}} \underbrace{h_u^{(l)}}_{\text{Message}}}_{\text{Aggregation}}) \tag{2.14}$$

where $h_v^{(l)}$ is the feature vector of node v at layer l, $\mathcal{N}(v)$ is the set of neighbors of node v, $\deg(v)$ is the degree of node v, $W^{(l)}$ is a learnable weight matrix, and σ is a non-linear activation function.

The term $\frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}}$ acts as a symmetric normalization, ensuring that the node degrees do not affect the scale of the feature representations. This formulation allows GCNs to effectively capture local graph structure and node features simultaneously [75].

GraphSAGE GraphSAGE [48] generalizes the notion of convolution to graphs by learning a set of aggregator functions that can be applied to any node's local neighborhood. It generates node embeddings by sampling and aggregating features from a node's neighbors. The general form of a GraphSAGE layer for a node v is:

$$h_v^{(k)} = \sigma(\underbrace{W^k}_{\text{Update}} \cdot [\underbrace{h_v^{(k-1)}}_{\text{Self-message}}, \underbrace{\text{AGG}_k(\{\underbrace{h_u^{(k-1)}}_{\text{Messages}}, \forall u \in \mathcal{N}(v)\})}_{\text{Aggregation}}]) \tag{2.15}$$

where $h_v^{(k)}$ is the embedding of node v at layer k, $\mathcal{N}(v)$ is the neighborhood of v, AGG_k is an aggregator function, W^k is a learnable weight matrix, σ is a non-linear activation function, and $[,]$ denotes concatenation.

The key innovation in GraphSAGE is the flexible aggregator function AGG_k, which can take on different forms depending on the chosen aggregation strategy [48]:

- Mean aggregator:

$$\text{AGG}_{\text{mean}} = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \tag{2.16}$$

  This aggregator computes the element-wise mean of the feature vectors of the neighboring nodes. It treats all neighbors equally, providing a simple yet effective way to summarize the local neighborhood structure. The mean aggregator is particularly useful when the importance of all neighbors is assumed to be uniform.

- Max-pooling aggregator:

$$\text{AGG}_{\text{max}} = \max(\{\sigma(W_{\text{pool}} h_u^{(k-1)} + b), \forall u \in \mathcal{N}(v)\}) \tag{2.17}$$

  The max-pooling aggregator applies a learnable linear transformation ($W_{\text{pool}} h_u^{(k-1)} + b$) to each neighbor's feature vector, followed by an element-wise max operation. This allows the model to capture different aspects of the neighborhood by learning to extract the most important features. The max operation provides permutation invariance, ensuring that the order of neighbors does not affect the result.

These aggregators allow GraphSAGE to adapt to different graph structures and learn more expressive neighborhood representations. The choice of aggregator can significantly impact the model's performance and its ability to capture different types of structural information in the graph.

Graph Attention Networks (GAT) GAT [132] introduces an attention mechanism to GNNs, allowing the model to assign different importance to different neighbors when aggregating information. The update rule for a GAT layer for node i is:

$$h_i^{(l+1)} = \sigma(\underbrace{\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{(l)}}_{\text{Aggregation}} \underbrace{W^{(l)}}_{\text{Update}} \underbrace{h_j^{(l)}}_{\text{Message}}) \tag{2.18}$$

where $\alpha_{ij}^{(l)}$ are attention coefficients computed using a self-attention mechanism:

$$\alpha_{ij}^{(l)} = \frac{\exp(LeakyReLU(a^T[Wh_i^{(l)} \| Wh_j^{(l)}]))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(LeakyReLU(a^T[Wh_i^{(l)} \| Wh_k^{(l)}]))} \tag{2.19}$$

The attention mechanism allows GAT to focus on the most relevant parts of the input graph for the task at hand. This is particularly useful in heterogeneous graph settings where different neighbor relationships may have varying levels of importance [132].

Each of these architectures represents a different approach to implementing the message passing mechanism introduced in Section 2.1.2. They have different strengths and are suited to various types of graph learning tasks. Understanding these architectures and their properties is crucial for analyzing their potential vulnerabilities to link stealing attacks, which we will explore in subsequent chapters.

2.2 Federated Learning

2.2.1 Introduction to Federated Learning

Federated Learning (FL) has emerged as an innovative machine learning paradigm that enables the training of models on distributed datasets without centralizing the data [87]. This approach addresses critical challenges of data isolation and privacy preservation in various domains, including scenarios involving graph-structured data [144].

FL offers several key advantages, as itemized below:

- **Decentralized data storage:** Raw data remains on local devices or servers, preserving data locality and ownership.
- **Collaborative model training:** Multiple parties can contribute to model improvement without directly sharing their data.
- **Privacy preservation:** Sensitive information is not explicitly exchanged between participants.
- **Reduced communication overhead:** Only model updates are transmitted, significantly decreasing data transfer requirements.

These characteristics make FL an attractive solution for scenarios where data cannot be centralized due to privacy concerns, regulatory restrictions, or

practical limitations. For instance, in healthcare, FL can enable collaborative research across institutions without sharing patient data directly.

In the context of graph-structured data, FL opens up possibilities for leveraging complex relational information across distributed networks while maintaining data locality [153]. This approach allows for the analysis of interconnected data, such as social networks or financial transaction graphs, without compromising the privacy of individual nodes or edges.

The application of FL principles to graph learning presents unique opportunities and challenges. It enables the development of techniques that can harness the power of collaborative learning while respecting the distinctive characteristics of graph data [52]. However, as we will explore more in Chapter 6, keeping data locally is not enough to mitigate against privacy attacks. We will see in Chapter 6 that we can exploit the communicated messages between the parties to construct a Link Inference Attack (LIA). Before discussing the privacy concerns of FL, we will first give an overview of how FL works, especially the Vertical Federated Learning (VFL) type applied to graph data, as it is the setting we worked with in our research [151].

2.2.2 Types of Federated Learning

FL can be categorized into several types based on how data is partitioned among participants [144]. The two main types are:

- Horizontal Federated Learning (HFL)
- Vertical Federated Learning (VFL)

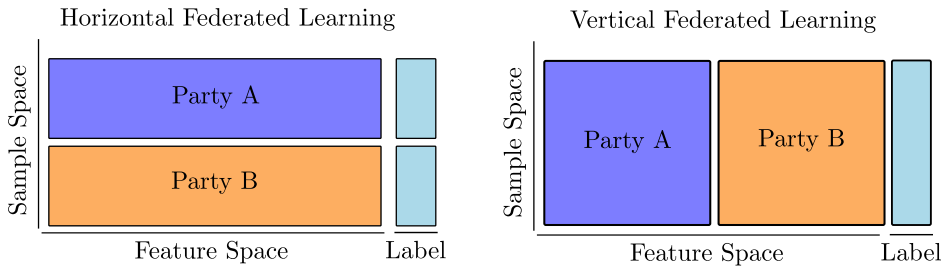


Figure 2.7: Comparison of Horizontal and Vertical Federated Learning (Inspired by [84])

Figure 2.7 illustrates the key differences between HFL and VFL. This visualization helps to clarify how data is distributed among parties in each type of federated learning.

Horizontal Federated Learning (HFL)

HFL, also known as sample-based federated learning, is applicable when different participants have the same feature space but different sample sets [87, 72]. As shown in the left part of Figure 2.7, in HFL:

- Participants share the same data structure but have different users or samples.
- Each participant's dataset can be considered as a horizontal partition of the complete dataset.
- Common in scenarios where multiple organizations have similar types of data for different users.

For example, two banks in different regions might have the same types of customer data (e.g., age, income, credit score) but for different sets of customers [144]. In the figure, we can see that Party A and Party B have the same feature space but different sample spaces, representing different user sets.

Vertical Federated Learning (VFL)

VFL, also referred to as feature-based federated learning, is used when participants have the same sample ID space but different feature sets [52, 84]. The right part of Figure 2.7 depicts VFL, where:

- Different participants hold different attributes or features for the same set of samples or users.
- The dataset can be viewed as being vertically partitioned across participants.
- Particularly useful when different organizations have complementary data about the same set of users.

An example of VFL could be a scenario where a bank and an e-commerce company collaborate, each having different types of data (financial and shopping behavior, respectively) about the same group of individuals [144]. In the figure, we can observe that Party A and Party B share the same sample space but have different feature spaces, indicating they possess different types of information about the same set of users.

VFL is particularly relevant to our research on federated graph learning, as it allows for the integration of graph structural information with additional node features that may be distributed across different parties [86, 14, 97].

In both HFL and VFL, as illustrated in the figure, the labels are typically held by a separate entity, often referred to as the active party or the central server. This party is responsible for aggregating the information from other parties and performing the final model training or prediction.

In the subsequent sections, we will focus more on VFL, especially its application to graph-structured data, as this forms the basis of our work on link inference attacks in vertical federated graph learning [151].

2.2.3 Vertical Federated Graph Learning (VFGL)

Building upon the concept of VFL, our research focuses on Vertical Federated Graph Learning (VFGL), which specifically deals with graph-structured data in a federated setting [151]. VFGL extends the principles of VFL to scenarios where one party owns a graph dataset, while another party holds additional features for the same set of nodes [86, 14].

VFGL System Architecture

In our VFGL setting, we consider a two-party scenario involving:

- $\mathcal{P}_\mathcal{G}$: Party owning the graph dataset $\mathcal{D}_\mathcal{G} = (\mathcal{G}, X_\mathcal{G})$
- $\mathcal{P}_\mathcal{A}$: Party holding a separate feature set $X_\mathcal{A}$
- $\mathcal{P}_\mathcal{Y}$: Active party owning the training labels $\mathcal{Y}$

The parties share a user space of N samples, where the graph $\mathcal{G}$ contains N nodes, each representing a user. $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$ manage user features of dimensions d_G and d_A respectively [97].

Figure 2.8 illustrates the VFGL system architecture in detail. This architecture demonstrates how different parties collaborate in the VFGL process while maintaining data privacy. Let's break down the components:

1. **Party $\mathcal{P}_\mathcal{G}$ (Graph Owner):** This party possesses both the graph structure $\mathcal{G}$ and associated features $\mathcal{F}_\mathcal{T}$. It uses a Graph Neural Network (GNN) to process this data, capturing both the structural and feature information of the graph.

2. **Party $\mathcal{P}_\mathcal{A}$ (Additional Feature Owner):** This party holds additional features $\mathcal{F}_\mathcal{A}$ for the same set of nodes. It employs a Deep Neural Network (DNN) to process these features.

3. **Party $\mathcal{P}_\mathcal{Y}$ (Label Owner):** This is the active party that owns the training labels $\mathcal{Y}$ and coordinates the learning process. It uses a DNN to aggregate information from other parties and produce the final output.

4. **Data Flow:**

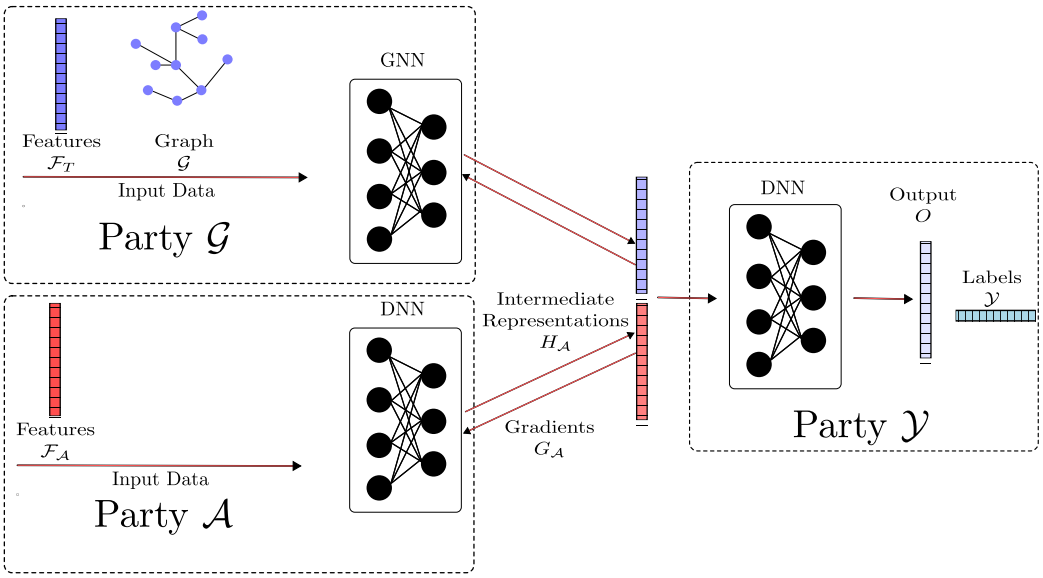


Figure 2.8: Vertical Federated Graph Learning (VFGL) System Architecture

- Parties $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$ process their respective data (graph and features) through their neural networks.
- The resulting intermediate representations $H_\mathcal{A}$ and $H_\mathcal{G}$ are sent to $\mathcal{P}_\mathcal{Y}$.
- $\mathcal{P}_\mathcal{Y}$ combines these representations in its DNN to produce an output O.
- Gradients $G_\mathcal{A}$ and $G_\mathcal{G}$ are then computed and sent back to parties $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{G}$ for model updates.

This architecture ensures that raw data (graph structure, features, and labels) never leaves its respective owners, preserving data locality while enabling collaborative learning.

Note on Scalability: While we describe a two-party scenario for simplicity, it is important to note that the VFGL framework can be extended to include multiple parties owning different feature sets [84, 144]. In practice, there could be several parties, each contributing unique features to the collaborative learning process. The architecture would then expand to accommodate these additional parties, with each new party following a structure similar to Party $\mathcal{P}_\mathcal{A}$ in the current setup.

VFGL Training Process

The VFGL training process involves:

- $\mathcal{P}_\mathcal{G}$ employing a GNN to transform $X_\mathcal{G}$ into intermediate representation $H_\mathcal{G}$
- $\mathcal{P}_\mathcal{A}$ using a DNN to transform $X_\mathcal{A}$ into intermediate representation $H_\mathcal{A}$
- $\mathcal{P}_\mathcal{Y}$ aggregating these representations and training a Deep Neural Network (DNN) for classification
- $\mathcal{P}_\mathcal{Y}$ computing the loss function $\mathcal{L}$ and gradient derivation
- $\mathcal{P}_\mathcal{Y}$ sending the gradients to $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$
- $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$ updating their models

The gradients are computed according to:

$$\nabla_{\theta_k}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_i \frac{\partial \mathcal{L}}{\partial H_{i,k}} \frac{\partial H_{i,k}}{\partial \theta_k} \tag{2.20}$$

where θ_k represents the model parameters, $H_{i,k}$ is the latent representation of the i^{th} sample computed by party $\mathcal{P}_k$, and $\frac{\partial \mathcal{L}}{\partial H_{i,k}}$ is the gradient of the loss function $\mathcal{L}$ with respect to $H_{i,k}$ for $k \in \{\mathcal{G}, \mathcal{A}\}$.

Algorithm 1 provides a detailed overview of this process.

2.3 Differential Privacy

Differential Privacy (DP) is a rigorous mathematical framework for quantifying and limiting privacy loss in statistical databases. It was originally introduced in [30] in the context of microdata, that is, databases containing records at the level of individuals. The primary goal of DP is to enable the release of aggregate information about a dataset while protecting the privacy of individual records within that dataset.

At its core, DP provides a formal guarantee: the output of a differentially private algorithm should not reveal the presence or absence of any specific record in the database, up to a small factor controlled by a privacy parameter ε. This parameter, often referred to as the *privacy budget*, allows for a quantifiable trade-off between privacy and utility [32]. A lower value of ε provides stronger privacy guarantees at the potential cost of reduced utility.

Algorithm 1 Two-Party Vertical Federated Graph Learning

Require: learning rates $\eta_{\mathcal{G}}$ and $\eta_{\mathcal{A}}$
Ensure: Trained model parameters $\theta_{\mathcal{G}}$, $\theta_{\mathcal{A}}$, ψ

Parties $\mathcal{P}_{\mathcal{G}}$, $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{Y}}$ initialize $\theta_{\mathcal{G}}$, $\theta_{\mathcal{A}}$, ψ.
for each training iteration $t = 1, 2, \ldots$ **do**
    In **parallel** do the following:
        Party $\mathcal{P}_{\mathcal{G}}$:
            Computes $H_{\mathcal{G}} = GNN(X_{\mathcal{G}}, \theta_{\mathcal{G}})$
            Sends $H_{\mathcal{G}}$ to party $\mathcal{P}_{\mathcal{Y}}$
        Party $\mathcal{P}_{\mathcal{A}}$:
            Computes $H_{\mathcal{A}} = DNN(X_{\mathcal{A}}, \theta_{\mathcal{A}})$
            Sends $H_{\mathcal{A}}$ to party $\mathcal{P}_{\mathcal{Y}}$
    End
    Party $\mathcal{P}_{\mathcal{Y}}$ computes the prediction output $P_{\mathcal{Y}} = DNN((X_{\mathcal{G}}, X_{\mathcal{A}}), \psi)$
    $\mathcal{P}_{\mathcal{Y}}$ updates $\psi^{t+1} = \psi^t - \eta_{\mathcal{G}} \frac{\partial \mathcal{L}}{\partial \psi}$
    $\mathcal{P}_{\mathcal{Y}}$ computes and sends the gradients $G_{\mathcal{G}} = \frac{\partial \mathcal{L}}{\partial H_{\mathcal{G}}}$ and $G_{\mathcal{A}} = \frac{\partial \mathcal{L}}{\partial H_{\mathcal{A}}}$ to $\mathcal{P}_{\mathcal{G}}$ and $\mathcal{P}_{\mathcal{A}}$, respectively.
    In **parallel** do the following:
        Party $\mathcal{P}_{\mathcal{G}}$:
            Computes $\nabla_{\theta_{\mathcal{G}}} \mathcal{L}$ with Equation 6.1
            Updates $\theta_{\mathcal{G}}^{t+1} = \theta_{\mathcal{G}}^t - \eta_{\mathcal{G}} \nabla_{\theta_{\mathcal{G}}} \mathcal{L}$
        Party $\mathcal{P}_{\mathcal{A}}$:
            Computes $\nabla_{\theta_{\mathcal{A}}} \mathcal{L}$ with Equation 6.1
            Updates $\theta_{\mathcal{A}}^{t+1} = \theta_{\mathcal{A}}^t - \eta_{\mathcal{A}} \nabla_{\theta_{\mathcal{A}}} \mathcal{L}$
    End
end for

Since its inception, DP has been adapted to various data structures and machine learning contexts. Notably, in the realm of graph data, researchers have proposed different notions of DP to protect edge and node privacy [77, 50]. These adaptations highlight the flexibility of DP in addressing privacy concerns across diverse data types and applications.

In this section, we will explore the fundamental concepts of DP, including its formal definitions and key mechanisms such as the Laplace and Gaussian mechanisms. We will then discuss important properties of DP, its application in machine learning, and the specific challenges it faces in this domain. Finally, we will examine DP in the context of graph data and its relationship to privacy attacks, setting the stage for the novel contributions presented in later chapters of this thesis.

2.3.1 Formal Definitions of Differential Privacy

To formalize the notion of differential privacy, we first need to define key concepts that underpin its mathematical framework. These definitions provide the foundation for quantifying and reasoning about privacy guarantees in data analysis.

Neighboring Databases

We begin with the concept of neighboring databases, which is crucial for understanding the scope of privacy protection in DP:

Definition 1 (Neighboring databases [30, 32])**.** Let $\mathcal{D}$ be the class of possible databases. Any two databases $D, D' \in \mathcal{D}$ that differ in one record are called *neighbors*. For two neighbor databases, the following equality holds:

$$d(D, D') = 1,$$

where d denotes the Hamming distance.

The significance of this definition lies in its role in quantifying the impact of individual records. By focusing on databases that differ by just one record, DP aims to ensure that the presence or absence of any single individual's data does not substantially affect the outcome of an analysis. This provides a strong privacy guarantee: an individual's participation in a dataset cannot be inferred from the results of a differentially private computation [32].

ε-Differential Privacy

Building on the concept of neighboring databases, we can now define ε-differential privacy:

Definition 2 (ε-Differential privacy [30, 32])**.** A randomized mechanism $\mathcal{M}$ satisfies ε-DP with $\varepsilon \geqslant 0$ if, for all pairs of neighboring databases $D, D' \in \mathcal{D}$ and for all measurable $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$,

$$\text{P}\{\mathcal{M}(D) \in \mathcal{O}\} \leqslant e^{\varepsilon}\, \text{P}\{\mathcal{M}(D') \in \mathcal{O}\}.$$

This definition formalizes the core principle of differential privacy. It states that the probability of any output occurring should be at most e^{ε} times more likely when using one database compared to its neighbor. The parameter ε, known as the privacy budget, controls the strength of this privacy guarantee. Smaller values of ε provide stronger privacy protection but potentially at the cost of reduced utility or accuracy in the analysis [32].

The significance of ε-DP lies in its ability to provide a formal, quantifiable measure of privacy that is independent of an adversary's background knowledge or computational power. It offers a worst-case guarantee, ensuring that even if an attacker knows all but one record in the database, they still cannot confidently determine whether that record was used in the computation [30].

(ε, δ)-Differential Privacy

In practice, a relaxation of ε-DP is often used, known as (ε, δ)-DP:

Definition 3 ((ε, δ)-Differential privacy [30, 32])**.** A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-DP with $\varepsilon, \delta \geqslant 0$ if, for all pairs of neighboring databases $D, D' \in \mathcal{D}$ and for all measurable $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$,

$$\text{P}\{\mathcal{M}(D) \in \mathcal{O}\} \leqslant e^{\varepsilon}\, \text{P}\{\mathcal{M}(D') \in \mathcal{O}\} + \delta.$$

This definition introduces an additional parameter δ, which allows for a small probability of violating the ε-DP guarantee. The significance of (ε, δ)-DP, also known as approximate differential privacy, is that it provides more flexibility in achieving privacy guarantees, often allowing for more practical implementations or improved utility in certain scenarios [32].

The δ parameter can be interpreted as the probability of a *catastrophic* privacy failure. It is typically chosen to be very small (e.g., $\delta \ll \frac{1}{n}$ where n is the number of records in the database) to ensure that the chance of a

significant privacy breach remains negligible. This relaxation is particularly useful for mechanisms like the Gaussian mechanism, which we will discuss in subsequent sections [32].

These definitions form the mathematical backbone of differential privacy, providing a rigorous framework for analyzing and designing privacy-preserving data analysis techniques. They allow us to reason precisely about privacy guarantees and trade-offs between privacy and utility in various data processing scenarios [30, 32].

2.3.2 Properties of Differential Privacy

Differential privacy possesses several important properties that make it a powerful and flexible framework for privacy-preserving data analysis. These properties allow for the composition of differentially private mechanisms and provide guarantees about post-processing of their outputs.

Composition Theorems

Composition theorems are fundamental to the practical application of differential privacy, as they allow us to reason about the privacy guarantees of complex algorithms that may involve multiple differentially private components.

Sequential Composition

Theorem 1 (Sequential Composition [33])**.** If each mechanism $\mathcal{M}_i$ in a k-fold adaptive composition $\mathcal{M}_1, \ldots, \mathcal{M}_k$ satisfies $(\varepsilon_i, \delta_i)$-DP, then the entire k-fold adaptive composition satisfies $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.

The significance of sequential composition lies in its ability to quantify the cumulative privacy loss when multiple differentially private mechanisms are applied sequentially to the same dataset. This theorem shows that the privacy guarantees *add up* in the worst case, allowing us to budget privacy loss across multiple computations.

Advanced Composition

While sequential composition provides a basic understanding of how privacy guarantees compose, advanced composition offers a tighter bound on the overall privacy loss, especially for large numbers of computations.

Theorem 2 (Advanced Composition [33])**.** If each mechanism $\mathcal{M}_i$ in a k-fold adaptive composition $\mathcal{M}_1, \ldots, \mathcal{M}_k$ satisfies (ε', δ')-DP for $\varepsilon', \delta' \geqslant 0$, then the entire k-fold adaptive composition satisfies $(\varepsilon, k\delta' + \delta)$-DP for $\delta \geqslant 0$ and

$$\varepsilon = \sqrt{2k \ln(1/\delta)}\varepsilon' + k\varepsilon'(e^{\varepsilon'} - 1). \tag{2.21}$$

The advanced composition theorem is particularly valuable when dealing with large numbers of computations, as it provides a sub-linear growth in the effective ε parameter. This allows for more queries or iterations in iterative algorithms while maintaining strong privacy guarantees.

Post-processing Property

Another crucial property of differential privacy is its resilience to post-processing:

Theorem 3 (Post-processing [33])**.** Let $\mathcal{M}$ be an (ε, δ)-differentially private mechanism and let f be an arbitrary function. Then $f \circ \mathcal{M}$ is (ε, δ)-differentially private.

The post-processing property ensures that any function of a differentially private output remains differentially private. This is significant because it allows for arbitrary data-independent transformations of the results of a differentially private computation without compromising the privacy guarantee. It means that once data has been processed in a differentially private manner, analysts can perform any further computations on the results without additional privacy concerns.

These properties collectively contribute to the power and flexibility of differential privacy as a framework for privacy-preserving data analysis. Composition theorems allow for the design of complex algorithms with manageable privacy loss, while the post-processing property ensures that the results of differentially private computations can be freely used in subsequent analyses. These characteristics make differential privacy well-suited for a wide range of applications in data science and machine learning, where multiple operations on sensitive data are often required [33].

2.3.3 Mechanisms for Achieving Differential Privacy

To implement differential privacy in practice, we need concrete mechanisms that satisfy the DP definition. Two of the most fundamental and widely used mechanisms are the Laplace mechanism and the Gaussian mechanism. Both

of these rely on the concept of global sensitivity.

Global Sensitivity

Before introducing the mechanisms, it is crucial to understand the concept of global sensitivity:

Definition 4 (L_p-**Global sensitivity** [32])**.** The L_p-global sensitivity of a query function $f\colon \mathcal{D} \to \mathbb{R}^d$ is defined as

$$\Delta_p(f) = \max_{\forall D, D' \in \mathcal{D}} \|f(D) - f(D')\|_p,$$

where D, D' are any two neighbor databases.

Global sensitivity quantifies the maximum change in the output of a function when applied to neighboring databases. This concept is fundamental to calibrating the noise added in differentially private mechanisms.

Laplace Mechanism

The Laplace mechanism is one of the simplest and most commonly used methods for achieving differential privacy:

Definition 5 (**Laplace mechanism** [32])**.** Given any function $f\colon \mathcal{D} \to \mathbb{R}^d$, the Laplace mechanism is defined as follows:

$$\mathcal{M}_L(D, f(\cdot), \varepsilon) = f(D) + (Y_1, \ldots, Y_d),$$

where Y_i are i.i.d. random variables drawn from a Laplace distribution with zero mean and scale $\Delta_1(f)/\varepsilon$.

Theorem 4 ([33])**.** The Laplace mechanism satisfies $(\varepsilon, 0)$-DP.

The Laplace mechanism achieves differential privacy by adding noise drawn from a Laplace distribution to the output of the query function. The scale of the noise is calibrated to the sensitivity of the function and the desired privacy level ε.

Gaussian Mechanism

While the Laplace mechanism provides pure ε-DP, the Gaussian mechanism is used to achieve (ε, δ)-DP:

Definition 6 (**Gaussian mechanism** [32])**.** Given any function $f : \mathcal{D} \to \mathbb{R}^d$, the Gaussian mechanism is defined as follows:

$$\mathcal{M}_G(D, f(\cdot), \varepsilon) = f(D) + (Y_1, \ldots, Y_d),$$

where Y_i are i.i.d. random variables drawn from a Gaussian distribution with zero mean and standard deviation $\Delta_2(f)\sqrt{2\log(1.25/\delta)}/\varepsilon$.

Theorem 5 ([33])**.** For any $\varepsilon, \delta \in (0, 1)$, the Gaussian mechanism satisfies (ε, δ)-DP.

The Gaussian mechanism adds noise drawn from a Gaussian (normal) distribution. It provides more flexibility than the Laplace mechanism, allowing for a trade-off between ε and δ in the privacy guarantee.

Both mechanisms illustrate the fundamental principle of differential privacy: achieving privacy by adding carefully calibrated noise to query results. The choice between Laplace and Gaussian mechanisms often depends on the specific requirements of the application, the desired privacy guarantees, and the nature of the query function.

Note: The concept of global sensitivity is crucial in both mechanisms. It determines the scale of noise added to achieve differential privacy. A higher global sensitivity requires more noise to be added to maintain the same level of privacy, potentially affecting the utility of the results. Therefore, designing queries or algorithms with low sensitivity is often a key consideration in practical applications of differential privacy.

2.3.4 Differential Privacy for Graphs

Applying differential privacy to graph data presents unique challenges due to the interconnected nature of graphs. Unlike traditional tabular data, where each record is independent, changes to a single node or edge in a graph can have far-reaching effects on graph statistics and learned representations. In the literature, we find two main approaches to adapt DP to graphs [77, 50], each offering different privacy guarantees and suited to different types of graph analyses.

Edge-level Differential Privacy

Edge-level DP focuses on protecting the privacy of individual connections in a graph.

Definition 7 (Edge-level adjacent graphs [77])**.** $\mathcal{G}$ and $\mathcal{G}'$ are considered *edge-level adjacent graphs* if one can be obtained from the other by removing a single edge. In other words, $\mathcal{G}$ and $\mathcal{G}'$ differ by at most one edge. Hence, their adjacency matrices differ by one element only.

Definition 8 ((ε, δ)-Edge-level differential privacy)**.** A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-edge-level DP with $\varepsilon, \delta \geqslant 0$ if, for all pairs of edge-level adjacent graphs $\mathcal{G}, \mathcal{G}'$ and for all measurable $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$,

$$\mathrm{P}\{\mathcal{M}(\mathcal{G}) \in \mathcal{O}\} \leqslant e^{\varepsilon}\, \mathrm{P}\{\mathcal{M}(\mathcal{G}') \in \mathcal{O}\} + \delta.$$

Edge-level DP is particularly useful when the privacy of individual relationships is the primary concern, such as in social network analyses or recommendation systems based on user interactions [106].

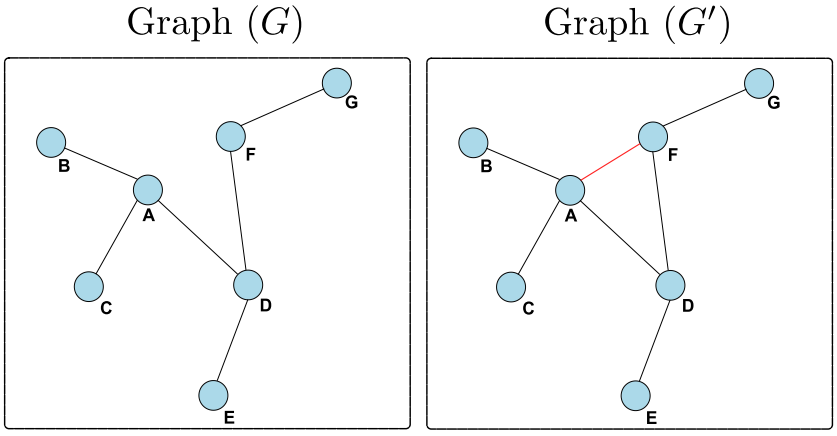


Figure 2.9: Illustration of edge-level differential privacy. The graphs G and G' are edge-level adjacent, differing only by the presence or absence of a single edge (highlighted in red).

Node-level Differential Privacy

Node-level DP provides a stronger privacy guarantee by protecting the presence or absence of entire nodes and their associated edges.

Definition 9 (Node-level adjacent graphs [50])**.** $\mathcal{G}$ and $\mathcal{G}'$ are said to be *node-level adjacent graphs* if one can be obtained from the other by removing a single node and all of its incident edges.

Definition 10 ((ε, δ)-Node-level differential privacy)**.** A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-node-level DP with $\varepsilon, \delta \geqslant 0$ if, for all pairs of node-level adjacent graphs $\mathcal{G}, \mathcal{G}'$ and for all measurable $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$, the following inequality holds:

$$\mathrm{P}\{\mathcal{M}(\mathcal{G}) \in \mathcal{O}\} \leqslant e^{\varepsilon}\, \mathrm{P}\{\mathcal{M}(\mathcal{G}') \in \mathcal{O}\} + \delta$$

Node-level DP is more suitable when the goal is to protect the participation of individuals in a network, rather than just their connections [15]. It provides a stronger privacy guarantee compared to edge-level DP, as it protects not only the relationships but also the presence or absence of individuals in the network.

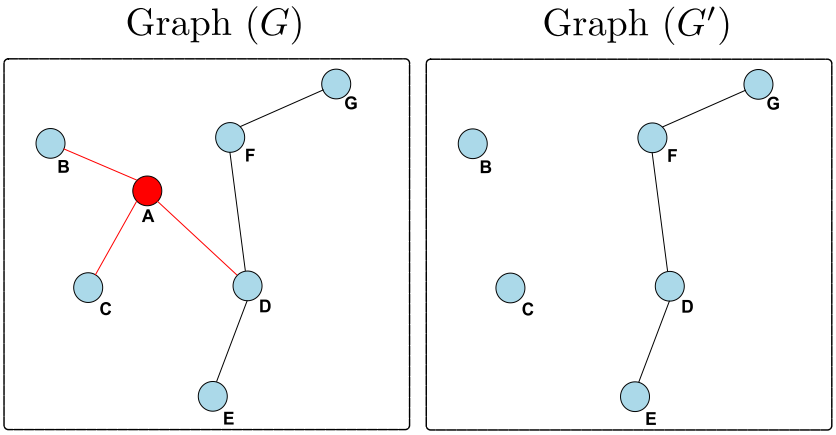


Figure 2.10: Illustration of node-level differential privacy. The graphs G and G' are node-level adjacent, differing by the presence or absence of a single node and all its incident edges (highlighted in red).

Applying DP to graphs presents unique challenges. The interconnected nature of graph data means that changes to a single node or edge can have far-reaching effects on graph statistics and learned representations. This interconnectedness often results in high global sensitivity for many graph algorithms, necessitating careful mechanism design to balance privacy and utility. Moreover, the choice between edge-level and node-level DP depends on the specific privacy requirements and the nature of the graph analysis

being performed [106, 15].

The application of differential privacy to graphs is particularly relevant in the context of Graph Neural Networks (GNNs), where the goal is to learn node or graph representations while preserving the privacy of the underlying graph structure. The challenges of applying DP to GNNs include maintaining the utility of learned representations while providing meaningful privacy guarantees, a topic that will be explored further in subsequent sections.

2.3.5 Differential Privacy in Machine Learning

Building upon the concepts of differential privacy in graphs, we now explore how DP is applied in the broader context of machine learning. The application of DP to machine learning algorithms aims to protect the privacy of individual data points in the training set while still allowing the model to learn useful patterns. There are three main approaches to incorporating DP into machine learning: input perturbation, objective perturbation, and output perturbation [67, 11].

Input Perturbation

Input perturbation involves adding noise to the training data before it is used in the learning algorithm.

Definition 11 (Input Perturbation)**.** Let D be the original dataset and f be a machine learning algorithm. Input perturbation creates a differentially private dataset D' by adding noise to D, and then applies f to D'.

This approach has the advantage of being algorithm-agnostic, as the privacy mechanism is applied independently of the learning algorithm. However, it may require significant amounts of noise to achieve strong privacy guarantees, potentially impacting the utility of the learned model [114].

Objective Perturbation

Objective perturbation modifies the objective function of the learning algorithm to incorporate differential privacy.

Definition 12 (Objective Perturbation)**.** Let $L(\theta; D)$ be the original loss function for a model with parameters θ on dataset D. Objective perturbation creates a new loss function $L'(\theta; D) = L(\theta; D) + Z(\theta)$, where $Z(\theta)$ is a carefully calibrated noise term.

This approach often allows for tighter privacy analysis and can lead to better utility compared to input perturbation for certain classes of learning algorithms [11]. In the context of GNNs, objective perturbation could involve adding noise to the loss function used for training the GNN, such as the cross-entropy loss for node classification tasks.

Output Perturbation

Output perturbation involves training the model on the original data and then adding noise to the resulting model parameters or predictions.

Definition 13 (Output Perturbation)**.** Let $f(D)$ be the output of a machine learning algorithm on dataset D. Output perturbation creates a differentially private output $f'(D) = f(D) + Z$, where Z is noise calibrated to the sensitivity of f.

This method can be computationally efficient, as it doesn't modify the training process. However, it may require careful analysis of the model's sensitivity to achieve meaningful privacy guarantees [139]. For a GNN, output perturbation could involve adding noise to the final node embeddings or to the weights of the trained GNN model.

These approaches to differential privacy in machine learning can be applied to various types of data, including graph data. The choice of method often depends on the specific machine learning task, the desired privacy-utility trade-off, and computational constraints. In the context of graph learning and GNNs, these methods need to be carefully adapted to account for the unique structure of graph data and the potential for privacy leakage through the graph topology [154].

It is important to note that the application of these DP approaches to graph data and GNNs presents additional challenges due to the interconnected nature of graphs. For instance, perturbing a single node or edge can have far-reaching effects on the learned representations of other nodes in the graph. This interconnectedness necessitates careful consideration of the privacy-utility trade-off and often requires the development of specialized DP mechanisms for graph learning tasks. The complexity of graph structures

makes it particularly challenging to balance the preservation of useful graph properties with the protection of individual privacy, a tension that continues to be an active area of research in the field of privacy-preserving graph analysis and learning.

2.4 Challenges of Differential Privacy in Machine Learning

While differential privacy provides a rigorous framework for privacy-preserving machine learning, its implementation comes with several challenges, particularly in the context of complex models and graph-structured data. We focus on three key challenges: the privacy-utility trade-off, parameter selection, and the impact on model performance.

2.4.1 Privacy-Utility Trade-off

The fundamental challenge in applying differential privacy to machine learning is balancing the need for privacy with the utility of the model. This privacy-utility trade-off refers to the inverse relationship between the strength of privacy guarantees and the usefulness of the data or model for analytical purposes. In the context of differentially private machine learning, stronger privacy guarantees (i.e., lower ε values) typically result in more noise being added to the data or model, which can degrade the model's performance [11].

This trade-off is particularly pronounced in graph-based models, where the interconnected nature of the data means that privacy-preserving noise can have cascading effects on the learned representations [154]. For instance, in the case of graph neural networks (GNNs), applying differential privacy mechanisms can lead to significant changes in the graph structure or node features, potentially altering important graph properties that the GNN aims to learn. As noted in [138], the privacy-utility trade-off in graph learning is more challenging due to the interdependence of data points in the graph structure.

2.4.2 Parameter Selection (ε and δ)

Choosing appropriate values for the privacy parameters ε and δ is crucial in differential privacy, but it remains a challenging task. The privacy budget ε quantifies the privacy loss, with lower values providing stronger privacy guarantees but typically at the cost of reduced utility. In (ε, δ)-differential

privacy, δ represents the probability of the privacy guarantee not holding and is typically chosen to be very small.

The selection of these parameters involves careful consideration of the specific application, the sensitivity of the data, and the desired level of privacy protection [138].

A significant challenge in parameter selection is the interpretation of ε values in real-world terms. While ε provides a mathematical bound on privacy loss, translating this into meaningful privacy guarantees for individuals or specific attack scenarios is not straightforward. This difficulty in interpreting ε has led to the development of empirical privacy measures. These measures involve evaluating the effectiveness of differential privacy mechanisms against specific privacy attacks, such as membership inference or reconstruction attacks [64]. By conducting these empirical evaluations, researchers and practitioners can gain a more concrete understanding of the privacy protection offered by a given ε value in the context of their specific data and use case.

In conclusion, while differential privacy offers a principled approach to privacy-preserving machine learning, its application, particularly in the context of graph-structured data and GNNs, presents significant challenges. Addressing these challenges requires a deep understanding of both the theoretical foundations of differential privacy and the specific requirements of graph-based machine learning tasks. Ongoing research in this area aims to develop more sophisticated privacy-preserving mechanisms that can better navigate the privacy-utility trade-off in complex, interconnected data structures.

Chapter 3

Related Work

The rapid integration of machine learning algorithms into privacy-sensitive domains has intensified concerns about data confidentiality and individual privacy. As models increasingly process and analyze sensitive information, understanding and mitigating potential privacy risks have become paramount. This chapter offers a comprehensive review of privacy attacks in machine learning, with a particular focus on membership inference attacks and link inference attacks, which are central to the discussions in this thesis.

We begin in Section 3.1 by exploring the taxonomy of privacy attacks in machine learning, categorizing their types and characteristics to set the stage for our specific investigations. This classification not only illuminates the landscape of privacy vulnerabilities but also contextualizes the threats addressed in our work.

Following this foundational overview, we delve into membership inference attacks in Section 3.2, tracing their evolution from initial applications in deep neural networks to their expansion into other machine learning models, such as PCA. We scrutinize defense strategies developed to counter these attacks, with a special emphasis on differential privacy as a principled and robust mechanism. This leads us to differentially private PCA, which underpins our exploration of membership inference attacks against PCA in Chapter 4.

We conclude the chapter by examining link inference attacks in Section 3.3, highlighting the unique privacy challenges posed by graph-structured data and graph neural networks. We analyze existing link inference attack methodologies and discuss their limitations, thereby setting the context for our contributions detailed in Chapters 5 and 6. Our analysis includes a review of defense mechanisms against link inference attacks, focusing on adapting differential privacy techniques to graph settings.

3.1 Privacy Attacks in Machine Learning

Over the past decade, machine learning algorithms have found application in a vast and rapidly growing number of systems for analyzing and classifying usually privacy-sensitive data. As these machine learning techniques are deployed in critical applications, they have also opened the door for potential attackers, raising significant privacy concerns. The increasing popularity of machine learning algorithms, including techniques such as PCA and GNNs, has led to a surge in research on their vulnerabilities [120, 138, 54].

Privacy attacks in machine learning aim to exploit vulnerabilities in machine learning models to extract sensitive information about the training data or the model itself. These attacks can be broadly categorized into four types:

- **Membership Inference Attacks (MIA)**: MIA aim to determine whether a particular data sample was part of the model's training dataset [120, 58, 17, 98, 150, 148].
- **Attribute Inference Attacks**: These attacks aim to infer sensitive attributes of individuals in the training data that were not explicitly included in the model's output [146, 39, 89, 65, 26].
- **Property Inference Attacks**: These attacks seek to infer some global properties of the training dataset that were not explicitly encoded in the model [41, 160].
- **Model Inversion Attacks**: In these attacks, adversaries attempt to reconstruct training data samples or extract sensitive features from the model [38, 24, 95].

While these attacks pose significant threats to privacy in general machine learning models, the rise of graph-structured data and GNNs has also introduced new privacy challenges specific to this domain. Graph data has become increasingly prevalent in today's data-driven landscape, particularly in applications involving social networks, biological systems, or recommendation engines [115]. However, these advantages in both efficiency and utility unfortunately come with a high cost in terms of privacy, as the underlying graph structure is usually considered sensitive information [9].

In the context of graph-structured data, additional privacy attacks have emerged:

- **Link Inference Attacks (LIA)**: LIA aim to discover relations among graph nodes by identifying or inferring whether or not there exist edges between them [54, 29, 138, 149, 151].

- **Property Inference Attacks**: These attacks aim to infer global properties of the target graph. Given the target graph embedding, the attack goal is to infer basic properties such as the number of nodes, the number of edges, or other structural characteristics of the graph [156, 134].

- **Subgraph Inference Attacks**: In these attacks, adversaries try to infer the existence of specific subgraph structures within the larger graph [156].

Among these various types of attacks, this thesis focuses primarily on two: MIA in the context of PCA, and LIA in the context of GNNs.

MIA are investigated specifically for PCA, extending the understanding of these attacks to unsupervised dimensionality reduction techniques. We study the impact of the attacks when the adversary has access to the principal components, proposing novel approaches to protect against such attacks in PCA scenarios.

LIA, on the other hand, are studied in the context of GNNs, where the graph structure itself is a key component of the data and model.

By focusing on these two types of attacks across different machine learning paradigms, our research aims to provide a comprehensive understanding of privacy vulnerabilities in both traditional machine learning techniques but non-investigated (PCA) and more recent graph-based approaches (GNNs).

3.2 Membership Inference Attacks

3.2.1 Foundations and Evaluation of MIA

Formal Description of MIA

MIA is formalized as a binary classification problem. Given a target model f trained on a dataset D_{train}, and an input data point x, the adversary aims to infer whether x is a member of D_{train} or not. Formally, the adversary's goal is to predict:

- **1**, if $x \in D_{\text{train}}$ (indicating membership).

- **0**, if $x \notin D_{\text{train}}$ (indicating non-membership).

The attack exploits the differences in the target model's behavior on training data versus non-training data, using model outputs such as predicted labels and confidence scores to make inferences.

Evaluation Metrics

The effectiveness of a MIA is assessed using several metrics. *True Positive Rate (TPR)* and *False Positive Rate (FPR)* are commonly used to measure the attack's ability to correctly identify members and the rate of incorrect member identifications, respectively. *Accuracy* provides an overall measure of the attack's performance. Additionally, *Area Under the Curve (AUC)* is particularly useful when the decision is threshold-based, and there is no prior knowledge about the optimal selection of the threshold. AUC reflects the attack's performance across various decision thresholds, giving a comprehensive evaluation of its effectiveness.

Attack Methodologies

MIA utilize various methodologies to infer membership. One common approach involves training *shadow models* [120] that mimic the behavior of the target model. These shadow models are trained on data from an auxiliary dataset that replicates the distribution of the target model's training data. The outputs of the shadow models, such as confidence scores [150] or predicted labels [17], are used to create an *attack model* that learns to distinguish between members and non-members based on these outputs. In scenarios where the adversary has *black-box access* [120, 94] to the target model, they rely on observing the model's responses to crafted queries. The success of these attacks often depends on the degree of overfitting in the target model, as well as the adversary's ability to simulate the target model's behavior effectively.

Literature Review

Since the introduction of MIA against deep neural network (DNN) models in [120], this attack has been extensively studied on DNNs and other ML models. The cited work formalizes the attack as a binary classification problem and trained neural network (NN) classifiers to distinguish between training members and non-members. The authors demonstrate that the main factor contributing to the success of membership inference attacks on DNN models is overfitting. Subsequent works [111, 85, 124, 66, 146] further develop MIA with different approaches against DNN of different architectures. The work in [124] reveals that by using suitable metrics, metric-based attacks result in similar attack performance when compared with NN-based attacks.

Besides DNN, membership inference attacks have also been investigated against logistic regression models [127, 109], k-nearest neighbors [127, 126], and decision tree models [127, 146]. Our work extends these studies to PCA.

As we shall elaborate later in Chapter 4, we propose, to this end, a novel MIA against PCA. To the best of our knowledge, there was no previous work trying to perform MIA on PCA.

Building upon our work on MIA against PCA, subsequent research has explored more advanced attacks. A recent study [79, 78] leverages our attack framework to develop a data reconstruction attack against PCA. This work demonstrates how an adversary can escalate from membership inference to reconstructing actual data points. This highlights the potential for more severe privacy breaches stemming from our initial MIA approach.

3.2.2 Defense Strategies for MIA

As MIA have become a significant privacy concern in machine learning, various defense strategies have been proposed to mitigate their impact. These strategies generally aim to reduce the model's susceptibility to overfitting or to obfuscate the model's outputs, making it harder for an attacker to distinguish between members and non-members of the training set.

Regularization

[116] is a fundamental approach in addressing model vulnerability to MIA. This technique involves adding a penalty term to the loss function during model training, which helps prevent overfitting, a key factor in model susceptibility to membership inference. L_1 and L_2 regularization are widely used methods that have demonstrated effectiveness in reducing a model's vulnerability to such attacks [120, 93]. Additionally, *dropout* [68], another form of regularization, has been shown to be particularly effective in mitigating MIA risks. By diminishing the model's capacity to memorize training data, these techniques obscure the patterns that MIA typically exploit to infer membership [74].

Data deduplication

is another crucial strategy in defending against MIA. This process involves removing duplicate or near-duplicate samples from the training data, which helps reduce the model's memorization of specific data points. Recent research has indicated that models trained on deduplicated datasets exhibit reduced vulnerability to the attack, underscoring the importance of unique data points in training robust models [111, 27]. By ensuring that models are not overly exposed to repetitive patterns, deduplication makes it more challenging for attackers to identify members of the training set.

Data augmentation

techniques, such as rotation, flipping, or adding noise to images, can artificially expand the training dataset. While these methods can increase the diversity of training data and help in generalizing the model's predictions, their effectiveness in the context of MIA defense is nuanced. Some forms of augmentation may inadvertently increase a model's vulnerability to the attack by introducing predictable patterns unless carefully managed [120, 74]. Therefore, while data augmentation can be beneficial, it requires careful implementation to ensure that it enhances rather than compromises privacy.

Output obfuscation

is yet another defense mechanism that focuses on modifying the model's output to make it less informative for attackers. Techniques such as rounding confidence scores or adding noise to the output have shown promise in mitigating MIA [64]. More advanced methods like MemGuard, which uses adversarial examples to obfuscate model outputs, have been demonstrated to effectively reduce the clarity of inference about whether specific data points were used in training [68]. However, it is important to note that while these methods can be effective, they often come with trade-offs in terms of model utility.

A more principled approach that has gained significant attention is the use of DP mechanisms. DP provides a formal framework for quantifying and limiting information leakage about individual training samples, making it particularly well-suited for defending against membership inference attacks. Unlike the aforementioned heuristic methods, DP offers provable privacy guarantees, albeit at the cost of some reduction in model utility.

The following subsections will delve deeper into the relationship between DP and MIA, exploring how DP can be effectively applied to protect against MIA. We will also examine specific applications of DP in the context of PCA. This exploration will provide insights into the practical implications of using DP as a defense mechanism against membership inference attacks in various machine learning contexts, and how it compares to the other defense strategies discussed in this section.

3.2.3 Differential Privacy and MIA

Differential Privacy (DP) and MIA are intrinsically linked in the landscape of privacy-preserving machine learning. At its core, DP aims to obfuscate the presence or absence of individual samples in a dataset, which directly

counters the objective of the attack—to determine whether a particular data point was used in training a model.

The fundamental promise of DP aligns with this goal: the output of a differentially private algorithm should be approximately the same regardless of the inclusion or exclusion of any individual's data in the input dataset. This property inherently limits the effectiveness of MIA by making it difficult for an adversary to distinguish between models trained on datasets that differ by only one record.

MIA serve as an empirical privacy measure, acting as an auditing tool to assess the practical effectiveness of DP protections. By attempting to infer membership, These attacks provide tangible evidence of a model's resilience against privacy breaches, complementing the theoretical guarantees offered by DP. Conversely, DP serves as a principled defense mechanism against the attack, providing a mathematical framework to quantify and limit information leakage about individual training samples.

In theory, there is a line of research aiming at establishing theoretical bounds on the success of membership inference attack in terms of DP privacy budgets. These works seek to quantify the relationship between the strength of DP guarantees (characterized by the privacy parameter ε) and the effectiveness of the attack. Notable contributions in this area include the work of [146], [35], and [61]. These studies provide increasingly refined bounds and insights into how DP parameters translate to protection against MIA, bridging the gap between theoretical DP guarantees and practical privacy protection in machine learning systems.

It is important to note that while theoretical bounds provide valuable insights, the practical application of DP against MIA often requires empirical evaluation. This is due to the differences in attack models and assumptions between DP (which considers worst-case scenarios) and practical attack implementations (which may operate under more limited adversarial knowledge). As highlighted in [64], these differences can lead to gaps between theoretical guarantees and empirical performance, emphasizing the need for comprehensive privacy assessment in real-world machine learning systems.

Differentially Private PCA

Differentially Private Principal Component Analysis (DP-PCA) has emerged as a crucial area of research in privacy-preserving machine learning, particularly in the context of defending against privacy attacks such as MIA. The goal of DP-PCA is to perform dimensionality reduction while providing rigorous privacy guarantees for the underlying data.

Several approaches to DP-PCA have been proposed in the literature. One

of the earliest works [34] introduces a method that adds Gaussian noise to the covariance matrix before performing eigen-decomposition. This approach, while simple, can significantly impact utility, especially for high-dimensional data. An alternative method [12] adds noise to the output of the algorithm rather than the input. This approach involves perturbing the top eigenvectors of the covariance matrix, which can lead to better utility in some scenarios.

The evolution of DP-PCA methods has seen various approaches. [6] introduces the SULQ framework which uses an input perturbation framework, the parameters of the noise are refined by [31]. [49] proposes a noisy power method that iteratively generates principal components while removing previously generated ones. [69] introduces a Wishart noise mechanism for covariance matrices, preserving positive semi-definiteness. Recent work [117] introduces two stochastic algorithms for differentially private PCA: DP-SPCA and DP-VRPCA. These methods employ gradient perturbation at each iteration, departing from previous power method-based techniques. The algorithms achieve tighter utility upper bounds with less noise, with DP-VRPCA showing better utility for large-scale datasets.

These studies collectively represent the evolution of various noise addition approaches in DP-PCA, including input perturbation, objective perturbation, and output perturbation methods, as explained in Section 2.3.5.

In the context of MIA, DP-PCA serves as a potential defense mechanism by introducing controlled noise into the PCA process. This noise makes it more difficult for an adversary to infer whether a particular data point was used in computing the principal components. However, the effectiveness of DP-PCA as a defense against MIAspecifically in the context of PCA had not been thoroughly explored in the existing literature.

Chapter 4 includes an investigation of how these existing DP-PCA algorithms can be leveraged to defend against MIA on PCA. In this chapter, we empirically evaluate the effectiveness of some of the above-mentioned methods, particularly [34], on mitigating our attack against PCA from an empirical privacy perspective. We study their privacy-utility trade-off, providing insights into the practical implications of using DP-PCA as a defense mechanism against the attack in the context of PCA.

3.3 Link Inference Attacks

3.3.1 Foundations and Evaluation of LIA

Formal Description of LIA

Link Inference Attacks (LIA) are formalized as a binary classification problem on graph-structured data. Given a target graph $G = (V, E)$ with nodes V and edges E, and a target model f (such as a Graph Neural Network, GNN) trained on G, the adversary aims to infer the presence or absence of an edge between two nodes $u, v \in V$. Formally, the adversary's goal is to predict:

- **1**, if $(u, v) \in E$ (indicating the presence of a link).
- **0**, if $(u, v) \notin E$ (indicating the absence of a link).

The attack exploits correlations between the model's outputs—such as node embeddings or prediction scores—and the underlying graph structure. By analyzing these outputs in response to various inputs, the adversary can infer sensitive relationships or interactions represented by the edges in the graph.

Evaluation Metrics

The effectiveness of a LIA is assessed using several metrics similar to those used in MIA. *True Positive Rate (TPR)* measures the attack's ability to correctly identify existing links, while *False Positive Rate (FPR)* indicates the rate at which the attack incorrectly infers links that do not exist. *Accuracy* provides an overall measure of the attack's performance in distinguishing between existing and non-existing links. Additionally, the *Area Under the Curve (AUC)* metric is particularly useful for threshold-based decisions, offering a comprehensive evaluation of the attack's effectiveness across various decision thresholds when the optimal threshold is unknown.

Attack Methodologies

LIA utilize various methodologies to infer link presence in graphs. A common approach involves analyzing the outputs of a target model, such as node embeddings or prediction scores, to detect patterns indicative of link existence. In cases where the adversary has *black-box access* to the target model, they can query the model with carefully crafted inputs and observe the outputs to infer links. Advanced attacks may involve manipulating node

features or injecting malicious nodes or edges to perturb the model's behavior, thereby revealing information about the graph's structure. The success of these attacks often hinges on the adversary's ability to exploit the model's sensitivity to changes in the graph and effectively simulate or influence the model's behavior.

Literature Review

The susceptibility of graph-based models to link inference attacks is initially demonstrated in [54], where correlations between node output predictions and the underlying graph structure are identified, revealing channels for edge information leakage. This work also examines how similarities in node features can be exploited to infer graph connections.

Building on this foundation, research in [28] shows that it is possible to reconstruct graph edges by analyzing predictions derived from node embeddings designed to capture graph topology. This study quantifies privacy leaks in graph embeddings and highlights the risks associated with releasing such embeddings.

More recently, the VertexSerum attack [25] introduces a graph poisoning strategy that enhances link-stealing efficacy by amplifying link connectivity leakage. This approach incorporates an attention mechanism within the link detection network, significantly improving the accuracy of node adjacency inference across various datasets and GNN architectures.

A notable advancement in this field is the LinkTeller attack [138], which demonstrates that an adversary can infer graph links by probing node features and analyzing their output predictions from a GNN, even without direct access to the graph structure. This attack is significant because it highlights the vulnerability of GNNs to inference attacks based solely on observable outputs and inputs.

However, the LinkTeller attack has several limitations. Firstly, when applied to discrete data, the attack's method of altering input features can convert them into continuous real values, making such perturbations more detectable and compromising the stealthiness of the attack in environments where data integrity is monitored. Secondly, the effectiveness of the LinkTeller attack diminishes when applied to deeper GNNs with more than three layers. The increased complexity and abstraction in deeper networks make it more challenging for the attack to extract meaningful link information. Additionally, the attack's performance can vary significantly across different graph structures and node feature distributions, which may limit its applicability in diverse real-world scenarios.

To overcome these limitations, a novel link inference attack named Node

Injection Link Stealing (NILS) is proposed in Chapter 5. NILS leverages the dynamic behavior of GNNs by injecting malicious nodes in a manner similar to adversarial attacks. This strategy demonstrates superior performance compared to existing attacks while providing a more systematic approach by exploiting the message-passing mechanism inherent to GNN architectures. By carefully designing the injected nodes and their connections, NILS effectively uncovers hidden links in the target graph without significantly altering the overall structure, thereby maintaining stealthiness and efficacy.

3.3.2 Link Inference Attacks in Federated Learning

In the realm of federated learning (FL), the primary objective is to mitigate privacy attacks on training data by keeping data localized to individual nodes. However, the exchange of information such as gradients and model outputs can still inadvertently leak sensitive information. Several studies have highlighted privacy vulnerabilities in FL settings. For example, one study demonstrates that gradients could reveal membership information about the training data [94]. Similarly, another study introduces gradient inversion attacks capable of reconstructing input data from gradients [162]. Moreover, research has shown that gradients might unintentionally disclose attributes unrelated to the main machine learning task, such as demographic information like gender or race [90].

Despite these findings, the specific privacy risks associated with the graph structure in FL have received limited attention, except in the context of Vertical Federated Learning (VFL) 2.2.2. In this setting, the first LIA is presented in [104], indicating that edge information could be inferred from intermediate representations and output predictions of the model. The study proposes multiple attack strategies based on the adversary's knowledge of the target graph. However, it does not explore potential leakages arising from gradients or the training labels of the data samples.

Building upon this prior work, a new LIA within the VFL framework is introduced in Chapter 6. This attack specifically investigates edge privacy leakage through gradients, aiming to enhance the understanding of edge privacy vulnerabilities in federated learning environments. Furthermore, analytical insights into the performance of LIA based on the properties of graph structures are provided, contributing to a more comprehensive understanding of privacy risks in graph-based federated learning.

The evolution of these attacks underscores the need for robust defense mechanisms to protect graph privacy. In the following subsection, various strategies proposed to defend against LIA are explored, discussing how they contribute to preserving the privacy of graph-structured data.

3.3.3 Defense Strategies for Link Inference Attacks

As link inference attacks pose significant privacy concerns in graph-based machine learning, various defense strategies have been proposed to mitigate their effectiveness. These strategies generally aim to protect sensitive link information either by obfuscating the graph structure and model outputs or by limiting the information available to potential attackers.

Random noise addition

is a fundamental approach to defending against link inference attacks. This technique involves introducing perturbations to various components accessible to potential attackers. For example, adding noise to learned node embeddings can help mask relationships between nodes, while perturbing model outputs can obscure inference patterns. However, these methods often face a challenging trade-off between privacy protection and model utility, as excessive noise can significantly impact the model's performance.

Output truncation

focuses on limiting the adversary's access to model outputs and intermediate representations. This approach includes restricting access to only predicted labels instead of complete model outputs, or limiting the information shared in intermediate representations [104]. While simpler to implement than other defense techniques, output truncation provides limited protection, as demonstrated in Chapter 6, where we show that even access to labels alone can reveal graph structure.

A more principled approach that has gained significant attention is the use of Differential Privacy for quantifying and limiting information leakage about graph structure, making it particularly well-suited for defending against link inference attacks. Unlike the aforementioned heuristic methods, DP offers provable privacy guarantees, albeit at the cost of some reduction in model utility.

The following subsection delves deeper into the application of differential privacy as a defense mechanism against link inference attacks, exploring how differential privacy can be effectively applied to protect graph privacy while maintaining useful learning capabilities.

Differential Privacy and Link Inference Attacks

Differential Privacy (DP) is a robust framework designed to protect individual privacy in data analysis by ensuring that the inclusion or exclusion of

a single data point does not significantly affect the output of an algorithm. In the context of graph data, DP serves as a powerful mechanism to mitigate privacy attacks such as LIA, where adversaries aim to infer sensitive relationships between nodes (edges). Similar to how DP safeguards against MIA —which attempt to determine whether specific nodes or data points are present in a dataset—DP can be extended to graphs to prevent adversaries from inferring connections between nodes.

In graph datasets, DP mechanisms have been developed to protect both node and edge information. Node-level DP as discussed in Section 2.3.4 focuses on preserving the privacy of individual nodes, preventing attackers from learning whether a specific node is part of the graph. Various DP mechanisms have been designed to achieve this, particularly in settings where membership inference attack poses a threat [8, 21]. These mechanisms ensure that attackers cannot reliably infer the presence of a node, offering strong protection against attacks that seek to expose sensitive information about specific entities within social networks or other graph-based systems [136, 137, 18].

While node-level DP effectively protects individual nodes, it is insufficient for safeguarding the relationships between nodes, which are often the target in LIA. Edge-level DP 2.3.4 addresses this specific challenge by ensuring that the presence or absence of a single edge between two nodes does not significantly influence the output of a graph analysis algorithm. This is particularly important in preventing LIA, where an adversary seeks to infer connections between specific nodes [77].

Research on edge-level DP has led to methods for publishing graph statistics under differential privacy guarantees, including subgraph count estimation [73] and degree distribution release [51, 22]. Although these statistics are useful for general graph analysis, they are often inadequate for training GNNs, which require access to the raw graph structure for the message-passing mechanism. Consequently, edge-level DP mechanisms have been extended to allow for perturbation of the graph structure while maintaining sufficient utility for machine learning tasks [138, 15, 96].

A notable example of an edge-level DP mechanism in graph data is the LapGraph algorithm [138]. LapGraph perturbs the adjacency matrix of a graph to achieve differential privacy. The algorithm operates by adding carefully calibrated noise to the adjacency matrix, ensuring that the perturbed graph preserves privacy while retaining structural properties necessary for analysis. The LapGraph algorithm splits the privacy budget into two parts: one for estimating the number of edges and another for perturbing the adjacency matrix. This process protects both the edge count and the overall structure of the graph under differential privacy, providing a solid defense

against LIA. It can be used as a preprocessing step before applying graph learning models like GNNs.

Algorithm 2 LapGraph Algorithm

Require: Original adjacency matrix A, privacy parameters ε_1, ε_2
Ensure: Differentially private adjacency matrix A'

$\Delta_1(A) \leftarrow 1$ ▷ L_1-sensitivity of adjacency matrix
$m \leftarrow$ number of edges in A
$\tilde{m} \leftarrow m + \text{Lap}(1/\varepsilon_1)$ ▷ Estimate number of edges
$N \leftarrow \max(\tilde{m}, 0)$ ▷ Ensure non-negative edge count
Generate Laplace noise matrix L with scale $\Delta_1(A)/\varepsilon_2$
$\tilde{A} \leftarrow A + L$ ▷ Add noise to adjacency matrix
Sort elements of $\tilde{A}$ in descending order
Set top N elements of sorted $\tilde{A}$ to 1, and remaining elements to 0
$A' \leftarrow \text{symmetrize}(\tilde{A})$ ▷ Ensure symmetry of adjacency matrix
return A'

When designing DP mechanisms, it is crucial to consider specific privacy threats and the capabilities of adversaries. For instance, the NILS attack discussed in Chapter 5 presents a scenario where an adversary can inject nodes connected to a targeted node and subsequently discover sensitive edge information, violating edge privacy. Traditional DP mechanisms may not sufficiently address this type of attack, and applying general DP mechanisms to protect against specific attacks can significantly reduce utility of the system. Instead, DP can be tailored or adapted for specific attack scenarios, as we explored in Chapter 5. In this context, the LapGraph mechanism could be adapted to help achieve our tailored DP notion. By adjusting the sensitivity in LapGraph, we can effectively protect against threats like NILS while maintaining the utility necessary for graph-based learning tasks.

Chapter 4

Privacy Considerations in Principal Component Analysis

4.1 Introduction

The rapid advancement and widespread adoption of machine learning (ML) algorithms have revolutionized data analysis across numerous domains in recent years. These algorithms have found applications in an ever-expanding array of systems designed to analyze and classify data, much of which is often privacy-sensitive [120, 111]. Among these ML techniques, Principal Component Analysis (PCA) stands out as one of the most commonly employed unsupervised learning algorithms. PCA's popularity stems from its ability to summarize the information content in large datasets by reducing their dimensionality while preserving as much variability as possible. The output of this statistical tool is a set of *principal components*, typically much smaller in number than the total attributes of the underlying data. This dimensionality reduction not only facilitates more efficient data processing and storage but also aids in visualization and feature extraction for downstream tasks [71]. However, the increasing ubiquity of ML algorithms, including PCA, has opened new avenues for potential privacy breaches, particularly when these techniques are deployed in critical applications handling sensitive data. In this chapter, we introduce and focus on a novel type of privacy attack: the *Membership Inference Attack* (MIA) against PCA. In our proposed attack model, an adversary intercepts the principal components computed over a dataset and attempts to infer whether a particular data sample was part of this dataset or not. We develop a membership prediction method that compares the reconstruction error - the distance between the original target sample and its PCA projection - against a predetermined threshold. This

approach represents a significant extension of MIA concepts, previously applied to other machine learning models [120, 146], to the domain of PCA. The implications of our proposed MIA against PCA can be severe, particularly in contexts where the mere association of an individual with a dataset can reveal sensitive information. For example, if an attacker can determine that a person's data was used in a PCA-based analysis of a particular medical condition, this alone could compromise the individual's privacy, regardless of what specific information about them was used. This scenario extends the privacy concerns previously identified in other domains, such as genomic studies [56], to PCA-based data analysis, highlighting the need for robust privacy protections in dimensionality reduction techniques. In this chapter, we conduct a comprehensive study of the effectiveness of MIAs against PCA and demonstrate that such attacks can achieve high performance, especially when the number of samples used by PCA is relatively small. Furthermore, to mitigate the risk posed by these attacks that exploit the leakage of principal components, we investigate the use of *differentially private* mechanisms. We evaluate how the privacy budget affects both the success rate of the attack and the utility of the PCA under differential privacy (DP). Our investigation makes several key contributions to the field of privacy-preserving machine learning:

1. We present, to the best of our knowledge, the first study on the impact of MIAs specifically targeting PCA, where the adversary has access to the principal components.

2. We propose and evaluate the use of differentially-private PCA algorithms as a countermeasure to MIAs, analyzing the impact of the privacy budget on both the utility of the PCA and the success rate of MIAs.

3. We provide a comparative analysis of different approaches to implementing differential privacy in PCA, including both vector and scalar queries under the so-called naive and advanced composition approaches.

4. We present experimental results comparing these different approaches under Gaussian and Laplace mechanisms for protecting PCA against MIAs.

The rest of this chapter is organized as follows: Section 4.2 describes our proposed membership inference attack against PCA, including the methodology and experimental results. Section 4.3 introduces differentially private PCA algorithms and evaluates their effectiveness in mitigating MIAs. Finally, Section 4.4 concludes the chapter with a summary of our findings and

their implications for privacy-preserving data analysis. Through this investigation, we aim to shed light on the potential privacy risks associated with PCA and provide insights into effective strategies for mitigating these risks while maintaining the utility of this valuable data analysis tool.

4.2 Membership Inference Attack against PCA

Building upon the foundations of PCA discussed in Section 2.1.1, we now delve deeper into the privacy implications of this widely used technique. As introduced in Section 2.1.1, PCA is primarily used for dimensionality reduction while preserving as much variability as possible in the data. However, this powerful technique can potentially lead to privacy vulnerabilities, particularly when applied to sensitive data.

In this section, we present a novel MIA targeting PCA. This attack aims to determine whether a specific data sample was part of the dataset used to compute the principal components, exploiting the concept of reconstruction error 2.1.1. We begin by examining how PCA's susceptibility to overfitting can lead to privacy risks, illustrated using the Olivetti faces dataset example from Section 2.1.1. Following this, we outline the threat model and attack methodology, providing a detailed description of our experimental setup. Finally, we present and analyze the results of our experiments, demonstrating the effectiveness of the attack under various conditions.

4.2.1 PCA Overfitting and Privacy Implications

Recall from Section 2.1.1 that the reconstruction error for a data point x_n in PCA is given by:

$$\mathcal{L}_n = \|x_n - \hat{x}_n\|_2^2 = \|x_n - WW^T x_n\|_2^2 \tag{4.1}$$

where $\hat{x}_n = WW^T x_n$ is the reconstruction of x_n after PCA, and W is the matrix of principal components.

While minimizing reconstruction error is crucial for PCA's effectiveness, it also poses privacy risks if PCA overfits the data. Overfitting is a common issue in many machine learning algorithms, including PCA, and it occurs particularly when the data set has fewer samples relative to its dimensionality. This can result in a model that captures specific data details, potentially exposing sensitive information.

To illustrate this concept, let us revisit the Olivetti faces dataset introduced in Section 2.1.1. Figure 4.1 shows the distribution of reconstruction errors for both training and test samples in this dataset.

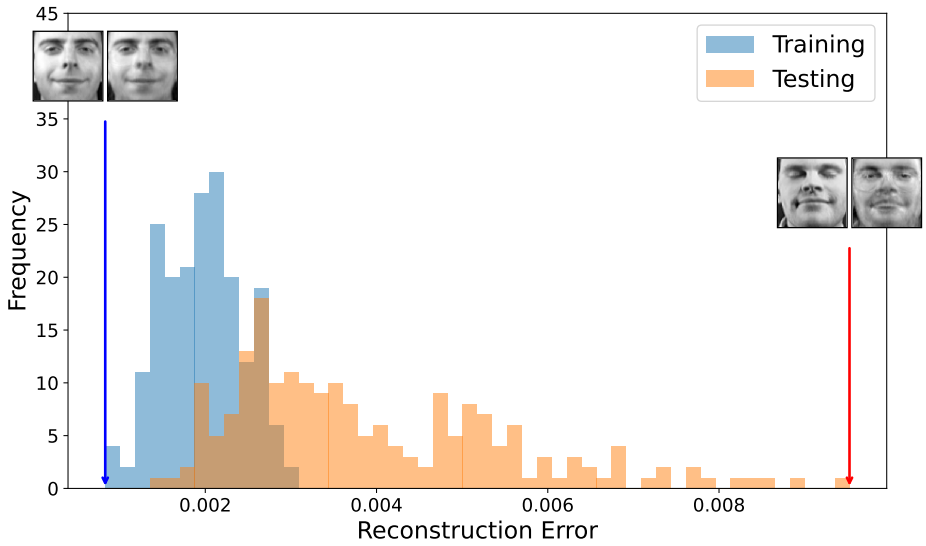


Figure 4.1: Histogram of reconstruction errors for training and test samples in the Olivetti faces dataset using 50 principal components. The inset images show comparisons of original (left) and reconstructed (right) face samples for both low and high reconstruction errors.

The histogram clearly demonstrates a significant difference between the reconstruction errors of training samples (blue) and test samples (orange). This visual representation supports the mathematical description of overfitting, which can be expressed as:

$$\frac{1}{|D_{train}|}\sum_{x\in D_{train}}\|x - WW^T x\|_2^2 \ll \frac{1}{|Dtest|}\sum_{x\in D_{test}}\|x - WW^T x\|_2^2 \tag{4.2}$$

where D_{train} and D_{test} represent the training and test sets respectively.

This overfitting in PCA raises significant privacy concerns, particularly the potential to identify whether a specific sample was part of the training set used to compute the principal components. Such capability could have severe privacy implications in various scenarios:

- In healthcare, identifying whether an individual's data was part of a PCA-analyzed study could reveal sensitive health information.
- In finance, determining if a particular transaction was in the PCA training set could expose confidential financial information.
- In social network analysis, inferring if a user's data was part of the PCA could reveal private social connections or behaviors.

The core of this privacy issue lies in the distinguishability between training and non-training samples based on their reconstruction errors (Equation 4.2). If an attacker can accurately estimate these errors, they might infer membership in the training set, compromising the privacy of individuals whose data was used to compute the PCA.

In our Olivetti faces example, the clear separation between training and test samples in Figure 4.1 illustrates this vulnerability. An attacker could potentially use this difference in reconstruction errors to determine whether a given face image was part of the original training set, thereby inferring an individual's participation in the face recognition system.

These scenarios underscore the importance of understanding and mitigating potential privacy risks when applying PCA to sensitive data. In the following sections, we will explore a specific type of privacy attack known as Membership Inference Attack (MIA) against PCA, and evaluate its effectiveness under various conditions.

4.2.2 Attack Methodology

Our attack scenario assumes that a curator computes the principal components V_k using a training dataset D and sends these components to a trusted party. The adversary $\mathcal{A}$ intercepts some or all of these components by eavesdropping on the communication channel. The goal of $\mathcal{A}$ is to identify whether a certain sample z is included in D, effectively discovering members of the training dataset.

This attack is particularly relevant in distributed settings [3], where multiple parties compute principal components of their individual (and usually smaller [63]) training datasets and send them to an aggregator. In such cases, $\mathcal{A}$ could compromise individual privacy by intercepting the principal components conveyed by each party.

To determine whether a sample z was used in computing the principal components, $\mathcal{A}$ employs the following strategy:

1. Compute the reconstruction error $\mathcal{L}(z, V_k)$ of the target sample z based on the intercepted first k eigenvectors, $V_k = [v_1, \ldots, v_k]$. Use the formula:
$$\mathcal{L}(z) = \|z - V_k V_k^T z\|_2^2 \tag{4.3}$$
2. Compare this error with a tunable decision threshold R.
3. If the reconstruction error is lower than the threshold, $\mathcal{A}$ predicts that z is a member of the training dataset D. Otherwise, $\mathcal{A}$ predicts that z is not a member of D.

The intuition behind this approach, as discussed earlier, is that samples from the training dataset are more likely to incur lower reconstruction errors compared to non-member samples. This is because PCA is designed to minimize the reconstruction error for the data it was trained on, as shown in Equation 2.5.

4.2.3 Experimental Setup

To evaluate the performance of our proposed attack, we conducted experiments using a diverse set of datasets. In this section, we describe the datasets used, the preprocessing steps applied, and the performance metrics employed to assess the attack's effectiveness.

Datasets

We selected four datasets for our experiments, representing different domains and data types:

- **UCI Adult** [5]: This dataset contains 48,842 records with 14 attributes, including both numerical (e.g., age, hours per week) and categorical (e.g., working class, education) features. We employed standard one-hot encoding to convert categorical attributes into numerical representations [62].

- **Census** [7]: This dataset comprises 1,080 records with 13 attributes of business statistics.

- **Labeled Faces in the Wild (LFW)** [60]: This dataset includes 13,233 images of 5,749 human faces collected from the web. The images have a resolution of 25×18 pixels. To balance the number of samples for each individual, we selected only one picture per person in our experiments.

- **MNIST** [81]: This dataset consists of 70,000 grayscale images of handwritten digits, each 28×28 pixels in size. It is commonly used for image classification tasks.

As a preprocessing step, we standardized all datasets to unit variance before conducting our attack experiments.

Performance Metric

To evaluate the success of our attack, we employed the Area Under the Receiver Operating Characteristic (ROC) curve (AUC) metric. The AUC indicates the relationship between true positive and false-negative rates over several decision thresholds R that the adversary can use to construct the attack. An AUC of 0.5 suggests that the attack performs no better than random guessing, while an AUC of 1.0 indicates perfect classification.

In all our experiments, we randomly selected equal-sized samples for both members and non-members. To ensure the robustness of our results, we reported the mean of the results over 10 trials.

4.2.4 Results and Analysis

We evaluated the success rate of our attack in terms of the number of principal components intercepted by the adversary, denoted by k. Figure 4.2 illustrates the maximum AUC that the adversary can achieve by observing the top-k principal components for various sample sizes N.

From our experimental results, we can draw several key observations:

1. **Impact of** k: The AUC increases with increasing k. This is intuitive, as the attacker gains access to more information with a larger number of principal components, enhancing their ability to distinguish between members and non-members.

2. **Impact of sample size**: The AUC decreases as the sample size N increases. This phenomenon can be attributed to the convergence of the sample covariance matrix A to the true covariance matrix of the dataset as N grows. This convergence makes the reconstruction errors of member and non-member samples increasingly indistinguishable. This behavior is analogous to what has been observed with neural networks when the training dataset is large [120].

3. **Dataset-specific performance**: For the MNIST and LFW datasets, the AUC is consistently above 0.5 and reaches 0.9 when $N = 1,000$. In contrast, the Census and Adult datasets yield much lower AUC values. This discrepancy can be primarily attributed to the smaller dimension d of these datasets. It's worth noting that MIAs against machine learning models trained on the Adult dataset have typically been unsuccessful in previous studies [120, 111].

These results demonstrate that our proposed MIA against PCA can be highly effective, especially when the number of samples used by PCA is small

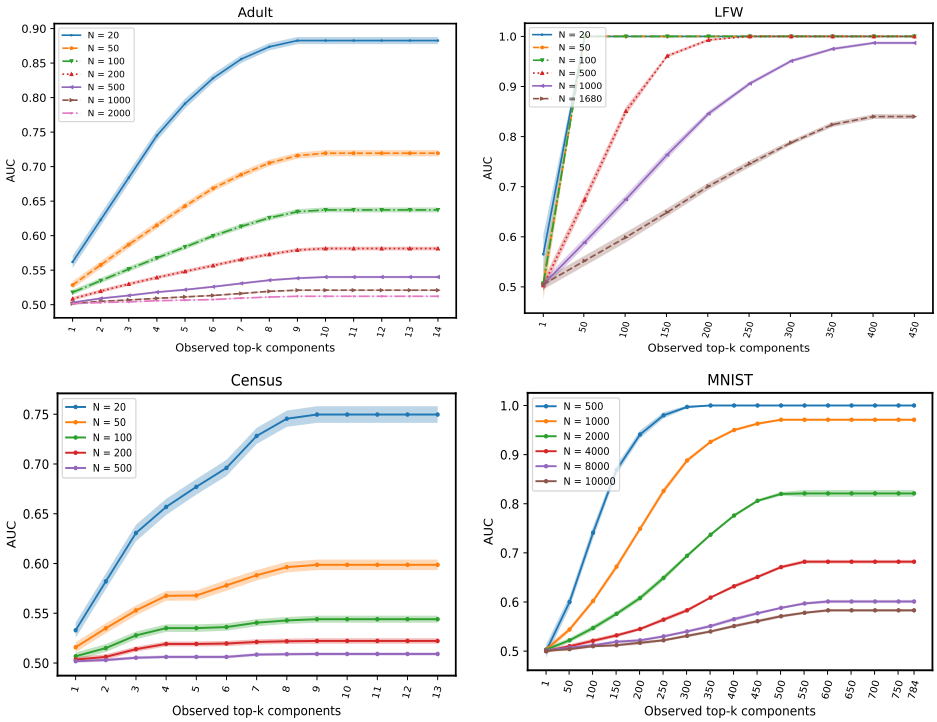


Figure 4.2: Impact of the sample size N and the observed top-k components on the attack's performance. Shaded areas show 95% confidence intervals for the mean.

and the dataset dimensionality is high. The attack's performance degrades as the sample size increases, which aligns with theoretical expectations and previous findings in the field of privacy-preserving machine learning.

Our findings highlight the potential privacy risks associated with sharing principal components, particularly in scenarios where the dataset size is limited or the data dimensionality is high. This underscores the need for privacy-preserving techniques when applying PCA in sensitive domains or distributed settings.

In the next section, we will explore how differential privacy can be applied to PCA to mitigate these privacy risks, and we will evaluate the trade-offs between privacy protection and utility preservation.

4.3 Differentially Private PCA

To mitigate the privacy risks associated with PCA, as demonstrated by the membership inference attack discussed in Section 4.2, we now turn our atten-

tion to differentially private PCA (DP-PCA) algorithms. These algorithms aim to protect individual privacy while still providing useful insights from the data.

In this section, we present and analyze four DP-PCA approaches: two based on the Gaussian mechanism and two on the Laplace mechanism. We evaluate their effectiveness in protecting against our MIA and assess the trade-off between privacy and utility.

4.3.1 DP-PCA Approaches

As in the previous scenario where no privacy protection was implemented, the first step for the data curator is to compute the principal components of the covariance matrix A, which are to be shared with a trusted entity. However, to protect individual privacy against an adversary who may intercept some or all components of A, the curator now decides to add Laplace noise directly to the coefficients q_{ij} of A. In the context of DP, this approach is called output perturbation.

To protect the $\alpha \doteq d(d+1)/2$ distinct coefficients of A (recall that A is a symmetric matrix), we consider two strategies: (i) using a joint query function that simultaneously queries all such coefficients, and (ii) querying each coefficient separately. We shall refer to these procedures as vector and scalar queries, respectively.

For $i = 1, \ldots, d$, let attribute i take values in the interval $[l_i, u_i]$ after standardization, and denote by Λ_i the absolute difference $|l_i - u_i|$. Recall [100] that $\Delta_1(q_{ij}) = \Lambda_i \Lambda_j / N$, from which we can easily derive an upper bound on $\Delta_1(A)$ just by adding up the sensitivities of all distinct coefficients. Accordingly, the scale of the Laplace noise injected to each coefficient yields $\Delta_1(A)/\varepsilon$ in the vector case, and $\Delta_1(q_{ij})/\varepsilon_{ij}$ in the scalar case, where ε is the total privacy budget and ε_{ij} the fraction thereof assigned to the coefficient q_{ij}.

Using the standard sequential composition property, we can compute the total privacy cost of the scalar strategy by adding up all ε_{ij} for $i \geqslant j$. In our experiments, in order to compare the two approaches for the same total privacy budget, we shall assume $\varepsilon_{ij} = \varepsilon/\alpha$. Note that, in this case, the noise scales will coincide only if $\sum_{i \geqslant j} \Lambda_i \Lambda_j = \alpha \Lambda^2$.

We now present four DP-PCA approaches that aim to protect individual privacy when computing and sharing principal components. These approaches differ in their noise injection mechanisms and how they compose multiple queries.

Laplace Mechanism-based Approaches

The Laplace mechanism adds noise drawn from a Laplace distribution to achieve differential privacy as described in Section 2.3.3. We consider two variations of this approach:

1. Laplace Vector Query This approach uses a joint query function that simultaneously queries all distinct coefficients of the covariance matrix A. The noise scale for each coefficient is:

$$\text{noise scale} = \frac{\sum_{i \geqslant j} \Lambda_i \Lambda_j}{N\varepsilon},$$

where N is the number of samples, and ε is the total privacy budget.

2. Laplace Scalar Query with Naive Composition This approach queries each coefficient of A separately. The noise scale for each coefficient q_{ij} is:

$$\text{noise scale} = \frac{\alpha \Lambda_i \Lambda_j}{N\varepsilon},$$

where ε/α is the privacy budget allocated to each coefficient.

Gaussian Mechanism-based Approaches

The Gaussian mechanism adds noise drawn from a Gaussian distribution to achieve differential privacy as described in Section 2.3.3.

3. Analyze Gauss (AG) Algorithm [34] This approach queries all coefficients of A simultaneously and estimates $\Delta_2(A)$ to be $1/N$. The noise scale for each coefficient is:

$$\text{noise scale} = \frac{\sqrt{2\log(1.25/\delta)}}{N\varepsilon}.$$

4. Laplace Scalar Query with Advanced Composition This approach uses the advanced composition theorem to achieve a tighter privacy guarantee when composing multiple differentially private mechanisms. The noise scale for each coefficient q_{ij} under this approach is:

$$\text{noise scale} = \frac{\Lambda_i \Lambda_j}{N\varepsilon'},$$

where ε' satisfies the advanced composition theorem equation for $k = \alpha$ and a given total privacy budget ε, δ.

Table 4.1 summarizes the four DP-PCA approaches and their respective noise scales.

Approach	Privacy notion	Noise scale
Laplace scalar query with naive composition	DP	$\alpha\Lambda_i\Lambda_j/N\varepsilon$
Laplace vector query	DP	$\sum_{i\geqslant j}\Lambda_i\Lambda_j/N\varepsilon$
Laplace scalar query with advanced composition	approx. DP	$\Lambda_i\Lambda_j/N\varepsilon'$
Analyze Gauss (AG) Algorithm [34]	approx. DP	$\sqrt{2\log(1.25/\delta)}/N\varepsilon$

Table 4.1: Overview of the DP mechanisms aimed to protect PCA against MIA. Here, ε denotes the *total* privacy budget and ε' the *fraction* thereof assigned to each coefficient of A.

4.3.2 Experimental Evaluation

We now present an experimental evaluation of the four DP-PCA approaches, focusing on their effectiveness in mitigating the proposed MIA and the trade-off between privacy and utility.

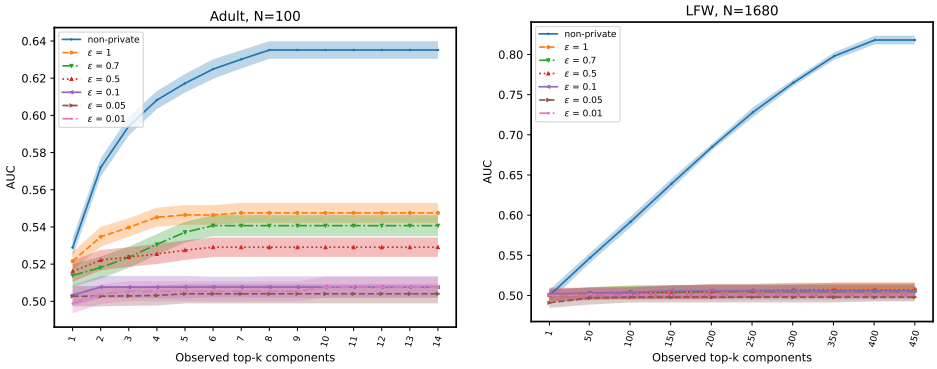
Protection Against MIA

To evaluate the protection provided by the DP-PCA approaches against membership inference attacks, we measure the Area Under the Curve (AUC) of the attack for various privacy budgets ε. Figure 4.3 shows the results for the AG algorithm and the Laplace vector query approach.

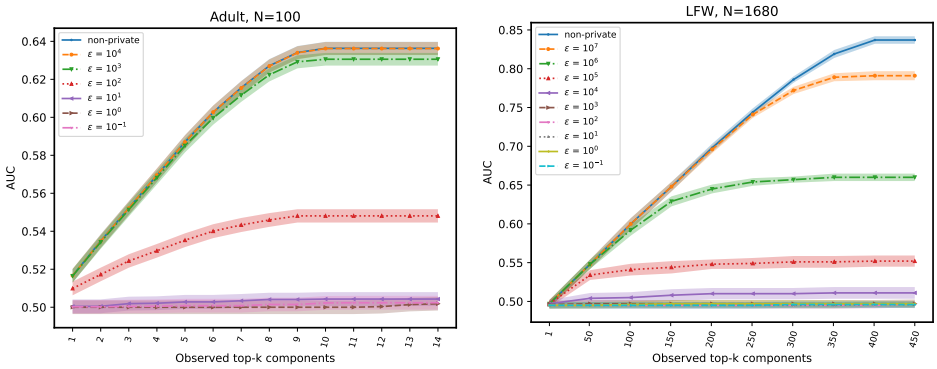
From Figure 4.3, we observe several key findings. The AG algorithm demonstrates robust protection against membership inference attacks, with AUC values consistently close to 0.5 (the random guess baseline) across all tested ε values. This indicates that the AG algorithm effectively obscures membership information, making it difficult for an attacker to distinguish between members and non-members of the dataset.

In contrast, the Laplace vector query approach exhibits a different behavior. As the privacy budget ε increases, we see a corresponding increase in AUC values. This trend suggests that the protection offered by this approach gradually diminishes as more privacy budget is allocated, eventually approaching the performance of the non-private case for large ε values.

Interestingly, the effectiveness of these approaches varies across different datasets. For the Adult and Census datasets, we find that the Laplace vector query approach with $\varepsilon = 10^2$ provides comparable protection to the AG algorithm with $\varepsilon = 1$. This equivalence in protection level occurs at different



(a) The AUC of the attack when the AG algorithm is applied with respect to k



(b) The AUC of the attack when the Laplace vector query algorithm is applied w.r.t. k.

Figure 4.3: The AUC of the attack when the AG algorithm (a) and the Laplace vector query approach (b) are applied with various values of ε. Shaded areas are the 95% confidence intervals for the mean.

privacy budgets for different datasets. For instance, in the case of the LFW dataset, the Laplace vector query approach requires a much larger privacy budget ($\varepsilon = 10^4$) to match the protection level of the AG algorithm with $\varepsilon = 1$.

These results underscore the capability of both approaches to significantly limit the success of membership inference attacks. However, they also highlight the superior performance of the AG algorithm, which maintains strong protection even at lower privacy budgets.

Comparison of Laplacian Approaches

Figure 4.4 compares the protection provided by the three Laplacian approaches (vector query, scalar query with naive composition, and scalar query

with advanced composition) for various levels of the total privacy budget.

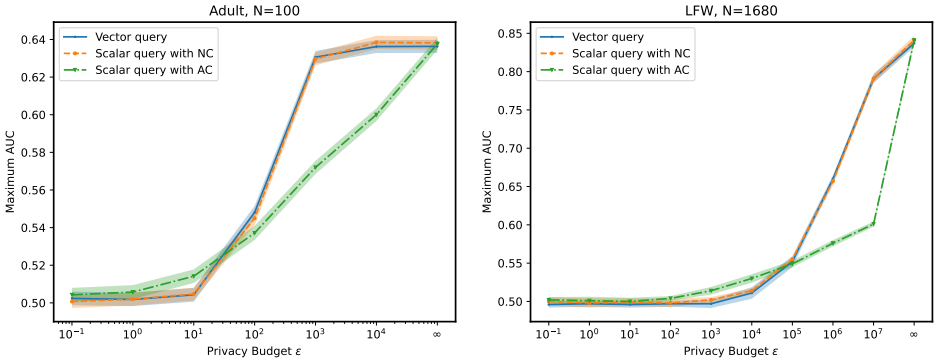


Figure 4.4: Attack performance with Laplacian approaches when the adversary intercepts all components ($k = d$). The infinity point represents the non-private case.

Analysis of Figure 4.4 reveals several important insights into the behavior of the Laplacian approaches. First, we observe that the advanced composition approach consistently achieves better protection than the naive approach in the low privacy regime (when ε is large). This superiority of the advanced composition method becomes particularly pronounced as the privacy budget increases.

The vector query and scalar query with naive composition approaches exhibit remarkably similar protection levels. This similarity can be attributed to their consumption of the same total privacy budget ε, resulting in comparable noise injection and, consequently, similar levels of privacy protection.

A notable feature in these results is the intersection point between the AUCs of the naive and advanced composition approaches. This intersection point, we find, is highly dependent on the dimensionality of the dataset. For datasets with lower dimensionality, such as Adult ($d = 14$, $\alpha = 105$) and Census ($d = 13$, $\alpha = 91$), the intersection occurs at a privacy budget of approximately $\varepsilon \approx 10^2$. In contrast, for higher-dimensional datasets like LFW ($d = 450$, $\alpha \approx 10^5$) and MNIST ($d = 784$, $\alpha \approx 3 \times 10^5$), this intersection point shifts significantly, occurring at a much higher privacy budget of $\varepsilon \approx 10^5$.

These differences can be explained by analyzing the noise scales of the naive and advanced composition approaches, as illustrated in Figure 4.5.

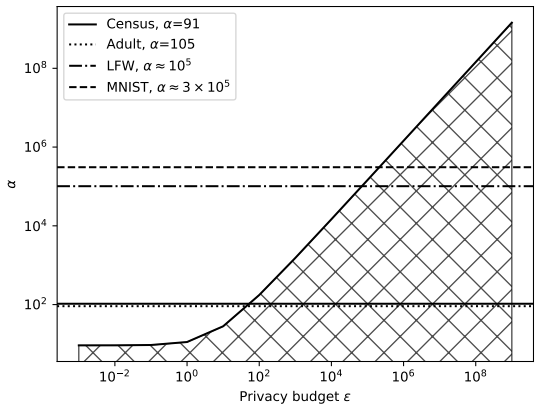


Figure 4.5: The hashed area shows where naive composition introduces less noise than advanced composition.

Trade-off Between Privacy and Utility

To evaluate the trade-off between privacy and utility for the DP-PCA approaches, we measure utility as the percentage of captured energy of the principal components produced by the DP-PCA algorithms, $\hat{V}_k$, with respect to the principal components of non-private PCA (SVD), V_k:

$$q = \frac{tr(\hat{V}_k^T A \hat{V}_k)}{tr(V_k^T A V_k)},$$

where A is the sample covariance matrix. For all datasets, we select the reduced dimension k such that V_k captures 90% of the energy.

Figures 4.6 and 4.7 show the utility of the DP-PCA algorithms as a function of the privacy budget ε and the AUC, respectively.

Analysis of Figures 4.6 and 4.7 reveals several crucial insights into the performance of the DP-PCA algorithms. The AG algorithm demonstrates particularly good utility for the Adult and Census datasets. However, its performance is notably lower for the other datasets examined. This disparity in performance across different datasets suggests that the effectiveness of the AG algorithm may be sensitive to certain dataset characteristics, a phenomenon that warrants further investigation.

In contrast, the Laplacian PCA solutions exhibit consistently lower utility compared to the AG algorithm when the privacy budget is small ($\varepsilon \leq 1$). This observation underscores the challenges of maintaining high utility while providing strong privacy guarantees, particularly in low-privacy-budget scenarios.

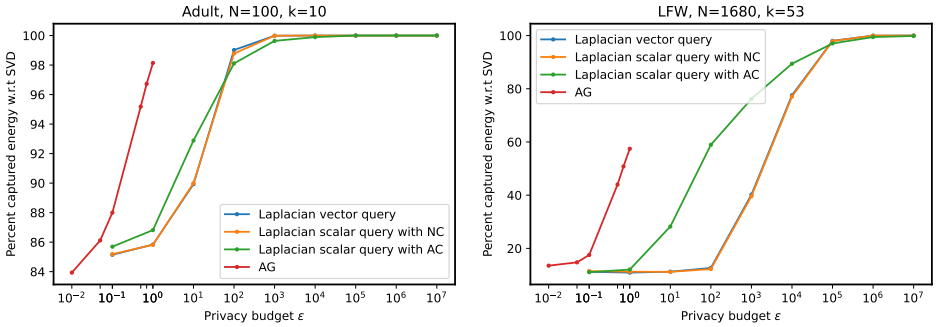


Figure 4.6: Trade-off posed by the four DP-PCA algorithms described in Section 4.3.1, between the total privacy budget ε and data utility. Utility is measured as the percentage of captured energy w.r.t. SVD.

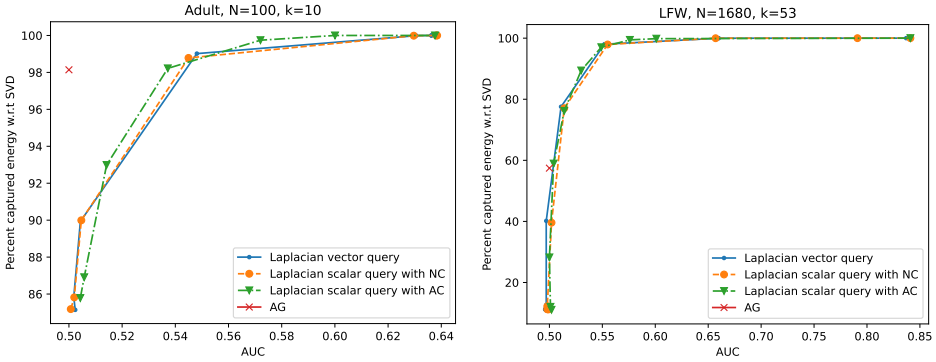


Figure 4.7: Trade-off posed by the four DP-PCA algorithms described in Section 4.3.1, between attack performance and data utility. We measure attack performance through AUC, and utility through the percentage of captured energy w.r.t. SVD.

Comparing the vector and scalar query approaches with naive composition, we find that they generally show similar utility. However, exceptions to this trend are observed in the MNIST and Census datasets, where the scalar query with naive composition achieves notably better utility. This discrepancy highlights the potential for dataset-specific optimizations in the choice of DP-PCA algorithms.

The advanced composition approach emerges as a promising method, providing better utility than naive composition under specific conditions. Specifically, this superiority is observed when the privacy budget ε and the number of queries α fall within the region depicted in the blank area of Figure 4.5. This finding suggests that the advanced composition approach may offer a more favorable trade-off between privacy and utility in certain scenarios.

4.4 Conclusion

In this chapter, we have conducted a comprehensive investigation into the privacy considerations surrounding Principal Component Analysis (PCA). Our study has shed light on the potential vulnerabilities of PCA to privacy attacks, specifically focusing on Membership Inference Attacks (MIAs). We have also explored the effectiveness of differentially private PCA algorithms as a defensive measure against such attacks.

Our main contributions in this chapter can be summarized as follows:

1. We implemented and evaluated the first membership inference attack against PCA, where an adversary has access to some or all principal components. This attack revealed significant privacy leakage in PCA, especially when the number of samples used is small.

2. We demonstrated that MIAs can be deployed successfully against PCA with high performance under certain conditions, highlighting the need for robust privacy-preserving techniques in PCA applications.

3. We evaluated the protection offered by differentially private PCA algorithms under various scenarios, including different protection algorithms, privacy budgets, numbers of intercepted principal components, and numbers of covariance coefficients.

4. We analyzed the trade-off between privacy protection and utility in differentially private PCA, providing insights into the practical value of privacy when these algorithms are employed.

Our findings indicate that the privacy risks associated with PCA are non-trivial and deserve serious consideration. The success of our MIA, particularly when the sample size is small, underscores the importance of implementing robust privacy-preserving techniques in PCA applications, especially those dealing with sensitive data.

The evaluation of differentially private PCA algorithms revealed that they can provide effective protection against MIAs, but at the cost of reduced utility. We observed that the choice of protection algorithm, privacy budget, and other parameters significantly influences both the level of privacy protection and the utility of the resulting PCA output.

This work contributes to the broader field of privacy-preserving machine learning by highlighting the privacy vulnerabilities in a fundamental technique like PCA and evaluating potential protective measures. Our results may be particularly useful for practitioners and researchers working with

sensitive data, helping them assess the practical value of privacy budgets when employing differentially private PCA algorithms in conjunction with desired utility levels.

Looking ahead, this research opens up several avenues for future work. One potentially fruitful direction would be to investigate whether there is a correlation between the samples vulnerable to privacy attacks in PCA and those in downstream tasks such as neural network classifiers. Such insights could lead to more comprehensive privacy-preserving strategies that protect data across multiple stages of machine learning pipelines.

Chapter 5

Node Injection Link Stealing Attack

5.1 Introduction

The rapid advancement of machine learning technologies has revolutionized data analysis across numerous domains. However, this progress has also given rise to significant privacy concerns, particularly when dealing with sensitive data. As explored in Chapter 4, even fundamental techniques like Principal Component Analysis (PCA) can be vulnerable to privacy attacks such as membership inference. Building upon these insights, this chapter delves into the privacy challenges specific to Graph Neural Networks (GNNs), a more complex and increasingly popular class of machine learning models.

Graph-structured data has become ubiquitous in today's data-driven landscape, finding applications in diverse fields such as social networks, biological systems, financial fraud detection, and recommendation engines. This data structure is particularly powerful because it captures not just individual entities (represented as nodes) but also the relationships between them (represented as edges). GNNs have emerged as state-of-the-art tools for analyzing and learning from such data, offering remarkable performance in various tasks including node classification, link prediction, and graph classification [140].

However, the very characteristics that make graph data so informative also make it highly sensitive from a privacy perspective. In social networks, for instance, the links between nodes typically represent shared interests, beliefs, political views, or even sexual preferences among users. The Cambridge Analytica scandal [9] starkly illustrated how this type of information can be exploited, potentially causing serious damage to individual privacy and even influencing democratic processes. Unlike in traditional machine

learning models where the privacy of individual data points is the primary concern, in graph-based models, the privacy of the relationships between data points becomes an additional critical consideration.

The privacy concerns surrounding graph-structured data and GNNs present unique challenges compared to those we encountered with PCA in the previous chapter. While PCA primarily deals with feature data, GNNs must protect both node features and the graph structure itself. This added complexity necessitates novel approaches to both attack and defend privacy in these models.

Motivated by these challenges, this chapter introduces a novel and powerful privacy attack against GNNs, which we call the Node Injection Link Stealing (NILS) attack. Our primary objective is to advance the understanding of edge privacy in GNNs by developing this attack and proposing a tailored Differential Privacy notion to protect against it. The NILS attack aims to infer the links among a set of target nodes in a graph, focusing on a specific scenario of training the GNN model for node classification tasks.

Previous works have explored various approaches to compromising privacy in GNNs. The Linkteller attack [138] demonstrated that by probing the features of nodes and analyzing the output predictions generated by the GNN, an attacker could successfully infer the links of the graph. Another study in [54] investigates the correlation of nodes' features to infer the links between them. However, these approaches have limitations. Linkteller's strategy of altering input features can be easily detected, especially when dealing with discrete data. The attack proposed in [54] requires access to a shadow dataset and the ability to train shadow GNNs, which may not always be feasible for an attacker.

Our proposed NILS attack takes a fundamentally different approach by exploiting the dynamic nature of GNNs. Unlike previous attacks, NILS adds a new node to the graph, connects it to the target node through a single edge, and further queries the model with malicious input features generated following different strategies. This approach allows the attacker to infer some of the target node's neighbors and hence steal a subset of the graph's connections.

The NILS attack conceptually mimics real-world actions such as sending a friend request on social media platforms, aiming to subsequently discern and analyze connections. This exploitation is akin to assessing changes in content recommendations or interactions that occur when a new connection is established. By leveraging this dynamic aspect of GNNs, NILS presents a more stealthy and potentially more effective attack vector compared to previous approaches.

Our contributions in this chapter can be summarized as follows:

1. We propose the NILS attack for inferring private links in a graph structure by injecting a new node, linking it to a target node, and employing various strategies to analyze the changes in the GNN's output.
2. We provide a comprehensive evaluation of the proposed attack's effectiveness on various datasets, demonstrating its superior performance compared to existing work such as LinkTeller [138] and link-stealing [54].
3. We explore the application of Differential Privacy (DP) mechanisms as a means to mitigate the effectiveness of our proposed attack, evaluating the trade-off between privacy preservation and model utility. To this end, we introduce a new notion of privacy specifically tailored to counter the NILS attack and evaluate defense strategies under this new notion.

These contributions not only advance our understanding of privacy vulnerabilities in GNNs but also provide insights that can be applied to other types of machine learning models dealing with relational data.

The rest of this chapter is organized as follows: Section 5.2 describes the threat model and details the methodology of our NILS attack. Section 5.3 outlines the experimental setup, including the datasets and models used in our evaluation. Section 5.4 presents the results of our experiments and provides an in-depth analysis of the attack's performance under various conditions. Section 5.5 introduces and evaluates our proposed defense strategy based on differential privacy. Finally, Section 5.6 concludes the chapter with a summary of our findings and their implications for privacy in GNNs, as well as discussing how these insights inform our subsequent investigations into privacy in federated learning settings, which will be explored in the following chapter.

Through this investigation, we aim to shed light on the potential privacy risks associated with GNNs and provide insights into effective strategies for mitigating these risks while maintaining the utility of these powerful learning models. By doing so, we contribute to the overall goal of this thesis: to comprehensively understand and address privacy challenges across various machine learning paradigms.

5.2 Attack Methodology

In this section, we present the methodology of our proposed Node Injection Link Stealing (NILS) attack. This attack exploits the dynamic nature of GNNs to infer private edge information within the graph structure.

5.2.1 Threat Model

We consider a scenario where a server has trained a GNN model using a specific dataset and offers access to this model through a black-box API. This API allows users to interact with the pre-trained GNN model without direct access to its internal components, such as the model architecture, parameters, or graph structure.

Users can submit prediction queries using node IDs. If a new node needs to be added to the graph, users can employ a *connect* query to attach the node to the graph before querying its prediction based on its ID. The API processes input data into output predictions, ensuring that the model's underlying computations remain hidden from the user.

5.2.2 Adversary's Goal and Knowledge

We consider an adversary, $\mathcal{A}$, who assumes the role of a GNN user. The objective of $\mathcal{A}$ is to determine the neighbors of a specific *target node*, v_t, selected from a set of *target nodes*, $V_{\mathcal{A}}$, within the graph. This is done based on the GNN's predictions for the node set $V_{\mathcal{A}}$. In simpler terms, $\mathcal{A}$ aims to identify the neighbors of the target node v_t that are included in the target set nodes $V_{\mathcal{A}}$.

The adversary $\mathcal{A}$ is able to obtain the predictions of the target nodes $V_{\mathcal{A}}$ by sending the server their corresponding IDs through the provided API. In addition, $\mathcal{A}$ is able to use the *connect* query to connect a node v_m to a target node v_t. In general, we assume that the adversary does not have access to the features of the nodes in the graph, with the exception of certain attack strategies described in Section 5.2.4.

5.2.3 Node Injection Link Stealing Attack

The NILS attack exploits the dynamic nature of the underlying GNN. The adversary $\mathcal{A}$ can *connect* new nodes and further query the prediction scores of a set of nodes $V_{\mathcal{A}}$ in the graph. While adding this new node v_m, $\mathcal{A}$ can choose which existing node v_t it actually connects to and hence try to discover its neighbors. The NILS attack consists of the following steps:

1. $\mathcal{A}$ first queries the prediction scores of the target nodes $V_{\mathcal{A}}$ and receives the corresponding prediction matrix P of the target nodes $V_{\mathcal{A}}$.

2. $\mathcal{A}$ generates malicious features of a malicious node v_m based on the obtained prediction matrix P (see Section 5.2.4 for further details on this step).

3. Next, $\mathcal{A}$ sends a *connect* query to inject the malicious node v_m. The query has the following parameters: the features x_m of the new node, and the ID of the target node v_t the adversary wishes to connect v_m to.

4. The server adds this malicious node v_m to the graph and links it to the target node v_t.

5. $\mathcal{A}$ queries back the server for new prediction matrix P' of the target nodes $V_{\mathcal{A}}$ and obtains it.

6. With access to P and P', $\mathcal{A}$ computes the L_1 distance between $P(v)$ and $P'(v)$ of each node v in $V_{\mathcal{A}}$.

7. A significant change in the prediction scores of a node v indicates a high probability of being a neighbor with v_t. If the difference exceeds a threshold R, the adversary infers that node v is a neighbor of v_t.

The decision threshold R is determined through an extensive parameter tuning process, aiming for an optimal trade-off between precision and recall in identifying the true neighbors of the target node. This balance is represented by the F_1 score. We evaluate various candidate values of R, selecting the one that yields the highest F_1 score as the optimal threshold.

Figure 5.1 illustrates the NILS attack strategy, depicting the interaction between the adversary and the server.

Algorithm 3 outlines the steps of the NILS attack.

5.2.4 Strategies for Generation of Malicious Node's Features

To evaluate how the injection of the malicious node v_m influences the predictions of the GNN, we study five strategies to generate the malicious node's features x_m. These strategies are designed with varying degrees of sparsity and stealthiness, enabling us to explore their effectiveness in altering the model's predictions. We define the proposed strategies as follows:

1. **All-ones strategy**: Generates a dense feature vector for the malicious node, containing all ones:

$$x_m = \mathbf{1}.$$

   This strategy potentially causes significant changes in predictions but may be less stealthy due to its dense feature vector.

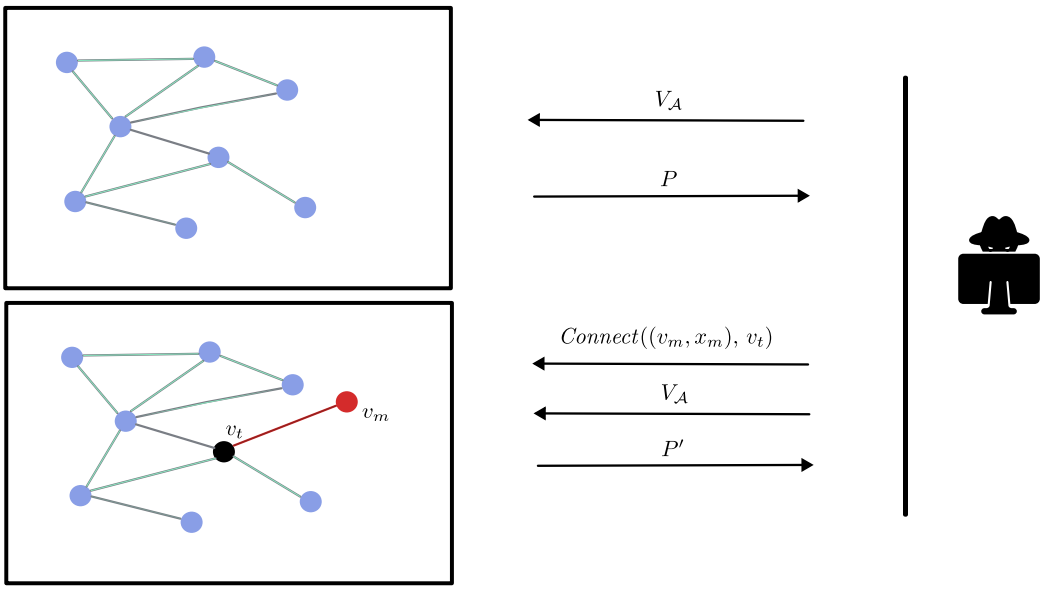


Figure 5.1: Adversary-Server Interaction: In the inference phase, the adversary first queries the prediction scores P of the target nodes, represented as $V_{\mathcal{A}}$. Next, the server sends the predictions P of the GNN to the adversary. Then, the adversary sends a *Connect* query to inject the malicious node v_m, with features x_m, to the target node v_t. Finally, after the injection, the adversary queries again the prediction scores P' of the target nodes $V_{\mathcal{A}}$.

2. **All-zeros strategy**: Creates a sparse feature vector for the malicious node, containing all zeros:

$$x_m = \mathbf{0}.$$

   This approach may subtly alter the output of the GNN, leading to smaller changes in predictions, while offering increased stealthiness.

3. **Identity strategy**: Introduces a malicious node with a feature vector identical to the target node's feature vector:

$$x_m = x_t.$$

   This strategy causes confusion in the model's predictions for neighboring nodes and has variable stealthiness based on the similarity between injected and target nodes. For this strategy, we assume that $\mathcal{A}$ knows the features of the target node x_t.

4. **Max attributes strategy**: This method creates a malicious node feature vector by computing the element-wise maximum of each attribute

Algorithm 3 Node Injection Link Stealing Attack

Require: set of nodes $V_{\mathcal{A}}$ and target node v_t
Ensure: the identified neighbors of v_t by the adversary

$P = \text{GNN}(V_{\mathcal{A}},\, X_{V_{\mathcal{A}}})$ ▷ Step 1
Generate malicious features x_m of node v_m ▷ Step 2
Connect node v_m to v_t ▷ Step 3-4
$P' = \text{GNN}(V_{\mathcal{A}} \cup v_m\ ,\, X_{V_{\mathcal{A}}} \cup x_m)$ ▷ Step 5
for each node v in $V_{\mathcal{A}}$ **do**
    $D(v) = \|P(v) - P'(v)\|_1$ ▷ Step 6
    if $D(v) \geq R$ **then**
        v is a neighbor of v_t
    else
        v is not a neighbor of v_t
    end if
end for

in the target nodes' feature matrix. Specifically, it considers only nodes from classes different from the target node's class:

$$x_{m,k} = \max_{i \in V_{\mathcal{A}},\ \text{with}\ C_i \neq C_t} X_{i,k}, \quad \text{for} \quad k = 1, \ldots, d.$$

Here, C_i represents the class of node i, and C_t is the class of the target node. This strategy potentially causes significant changes in predictions but may be less stealthy due to exaggerated features. We assume in this strategy that the adversary has access to the features of the set of target nodes $V_{\mathcal{A}}$ as well as to their predicted classes by the GNN.

5. **Class representative strategy**: This approach generates a malicious node feature vector by selecting the feature vector of the node with the highest confidence score for a specific class, different from the target node's class:

$$x_m = x_{i^*} \text{ with } i^* = \arg\max_{\substack{i \in V_{\mathcal{A}}, \\ C_i \neq C_t}} p_{i,j}.$$

In this equation, x_m is the malicious node feature vector, i^* is the node index with the highest confidence score for a specific class different from the target node's class, $V_{\mathcal{A}}$ is the set of target nodes, C_i represents the class of node i, and C_t is the class of the target node. This strategy leverages the model's predictions to alter the neighbors of the target node predictions, potentially offering increased stealthiness.

Additionally, we introduce the LinkTeller **Influence** strategy as an alternative to the original method in [138], incorporating their feature perturbation strategy. This strategy entails perturbing the features of the target node by adding a small real value α:

$$x_m = x_t + \alpha.$$

We assess the performance of the Influence strategy in comparison to other strategies, aiming to determine whether the attack performance gains are attributable to node injection or the crafting of malicious features. It is worth noting, however, that the Influence strategy may be easily detected if the feature x_t has a discrete nature, given that x_m is real-valued.

5.3 Experimental Setup

This section outlines the experimental setup used to evaluate our proposed NILS attack, including the datasets, models, and evaluation methodology.

5.3.1 Datasets

To evaluate the effectiveness of NILS in various contexts, we conducted experiments on several datasets:

- **Flickr** [152]: Nodes represent images on the Flickr platform. Edges connect nodes with shared properties (location, gallery, or comments). Node features are word representations derived from the images.
- **Twitch datasets** [107]: We use TWITCH-FR and TWITCH-RU for evaluation, and Twitch-ES for training GNNs in the inductive setting [138]. These datasets represent user follow connections. The task is binary classification of streamers' language use, based on features like preferred games, location, and streaming habits.
- **Citation networks** [76]: For the transductive setting, we use Cora, Citeseer, and Pubmed. These datasets represent citation relationships among publications. The task is to predict publication topics based on textual features.

This diverse selection of datasets allows us to evaluate NILS in both inductive and transductive settings across a range of application domains, providing a comprehensive assessment of the attack's effectiveness and generalizability.

5.3.2 Models

Our study follows the approach outlined in [138] for model training and hyperparameter selection. We trained Graph Convolutional Networks (GCNs) using various configurations and hyperparameters. These configurations encompassed different normalization techniques applied to the adjacency matrix, varying numbers of hidden layers (ranging from two to four), different input and output units, and various dropout rates.

To identify the optimal set of hyperparameters, we employed a grid search strategy. This involved systematically exploring combinations of hyperparameters and evaluating their performance on a validation set. The specific search space for hyperparameters and the formulae for different normalization techniques are detailed in [138, Appendix F].

After determining the best set of hyperparameters, we trained the GCN models to minimize the cross-entropy loss for the intended tasks. By utilizing the same training procedures and hyperparameter tuning strategies as in LinkTeller [138], we aimed to provide a comprehensive understanding of the attack performance across different layer configurations while maintaining consistency with previous work.

5.3.3 Evaluation Methodology

Our evaluation methodology was designed to provide a thorough and fair assessment of the NILS attack. In accordance with the methodology presented in [138], we employed precision, recall, and the F_1 score as our primary evaluation metrics. These metrics are particularly suitable for addressing the imbalanced nature of our binary classification problem, where the minority class (i.e., connected nodes) is of central interest.

For target node selection, we primarily chose a set of 500 target nodes ($|V_{\mathcal{A}}| = 500$) using a uniform random sampling approach. Furthermore, following the baseline study's example [138], we explored scenarios where target nodes exhibit either low or high degrees. This allowed us to assess the attack's performance across different node connectivity levels. A comprehensive discussion of the sampling strategy can be found in [138, Section V.D.].

To ensure the reliability of our results, we conducted three runs using different random seeds for each experiment, reporting the average results along with the corresponding standard deviations. For the evaluation of defense strategies, we increased the number of runs to five, again using different random seeds for LapGraph, and reported the averaged results.

We compared the performance of NILS with two baseline approaches:

1. The LinkTeller attack [138]

2. The link-stealing attacks introduced in [54]

For these comparisons, we used identical experimental setups where applicable to ensure fair evaluation.

To understand the influence of GNN architecture on the attack's success, we evaluated NILS against GNNs with varying depths (two, three, and four layers). This analysis provides insights into the relationship between model complexity and vulnerability to our proposed attack.

Finally, we assessed the effectiveness of our proposed defense strategy based on differential privacy. This evaluation involved applying the LapGraph algorithm [138] to achieve desired DP guarantees under our newly proposed one-node-one-edge-level DP notion. We measured both the attack's F_1 score and the classification task's F_1 score for the GCN across various privacy budgets ε.

To determine the optimal threshold R for inferring neighbors in our attack, we conducted a parameter tuning process. We evaluated various candidate values of R, selecting the one that yielded the highest F_1 score as the optimal threshold. This process ensures that our attack achieves the best possible balance between precision and recall in identifying true neighbors of the target node.

This comprehensive experimental setup allows us to thoroughly evaluate the effectiveness of our proposed NILS attack across various datasets, model configurations, and attack scenarios. In the following section, we present the results of these experiments and provide a detailed analysis of the attack's performance and its implications for privacy in GNNs.

5.4 Results and Analysis

This section presents the results of our experimental evaluation of the Node Injection Link Stealing (NILS) attack and provides an in-depth analysis of its performance under various conditions. We begin by examining the effectiveness of different strategies for generating malicious node features, then compare NILS to existing baseline attacks, and finally investigate the impact of GNN depth on the attack's success.

5.4.1 Analysis of Malicious Feature Generation Strategies

We evaluated five different strategies for generating the features of the malicious node v_m, as described in Section 5.2.4. Table 5.1 presents the F_1 scores achieved by each strategy across three datasets: Twitch-FR, Twitch-RU, and Flickr.

Method	Twitch-FR	Twitch-RU	Flickr
Class Rep.	0.94 ± 0.01	0.83 ± 0.06	0.96 ± 0.06
Max Attr.	0.99 ± 0.00	0.98 ± 0.02	$\mathbf{1.00 \pm 0.00}$
All-ones	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.97 \pm 0.01}$	0.99 ± 0.02
All-zeros	0.58 ± 0.02	0.48 ± 0.01	0.71 ± 0.07
Identity	0.81 ± 0.02	0.69 ± 0.01	0.95 ± 0.07
Influence NILS	0.81 ± 0.02	0.70 ± 0.01	0.89 ± 0.10
Influence LinkTeller [138]	0.80 ± 0.02	0.74 ± 0.01	0.32 ± 0.13

Table 5.1: F_1 scores and standard deviations for different attack methods and datasets.

The results reveal that the All-ones, Max attributes, and Class representative strategies are the most effective in causing significant changes in the predictions of the target node's neighbors. These strategies consistently achieve high F_1 scores across all datasets, with the Max attributes strategy performing exceptionally well on the Flickr dataset, achieving a perfect F_1 score of 1.00.

Conversely, the All-zeros and Identity strategies exhibit relatively lower success rates. While these strategies offer certain benefits in terms of stealthiness, their impact on the graph structure and predictions is less pronounced, highlighting a trade-off between attack effectiveness and stealthiness.

The Influence strategy, when implemented within our NILS framework, shows a modest improvement over the LinkTeller baseline for the Twitch-FR dataset. This suggests that the node injection property of our NILS attack is effective in this context. However, for the Twitch-RU dataset, NILS underperforms in comparison to the LinkTeller baseline when using the Influence strategy.

The most significant improvement is observed in the Flickr dataset, where the node injection property of NILS considerably increases the F_1 score from 0.32 ± 0.13 (LinkTeller) to 0.89 ± 0.10 (NILS with Influence strategy). This outcome highlights the advantage of the NILS attack's node injection

method, particularly when compared to the LinkTeller attack, which employs the Influence strategy without node injection.

Overall, the Max attributes approach significantly enhances the F_1 score of the attack beyond the baseline established by LinkTeller [138]. Specifically, for the Twitch datasets, on average, our method improves the F_1 score by 23.75%. For the Flickr dataset, it records a remarkable improvement, increasing the F_1 score from 0.32 to 1.0, an increase of 212.5% over LinkTeller [138].

These findings underscore the importance of considering both the effectiveness and stealthiness of malicious feature generation strategies when devising link inference attacks on GNNs. The choice of strategy can significantly impact the attack's success rate and its potential for detection.

5.4.2 Comparison with Baseline Attacks

We compared the performance of NILS with two baseline approaches: the LinkTeller attack [138] and the link-stealing attacks introduced in [54]. Table 5.2 presents the results of this comparison for the Twitch and Flickr datasets, while Table 5.3 shows the comparison for the citation network datasets.

The results in Table 5.2 demonstrate that NILS consistently outperforms LinkTeller on both Twitch datasets (TWITCH-FR and TWITCH-RU). Furthermore, NILS exhibits a substantial improvement over LinkTeller on the Flickr dataset, achieving nearly double the precision and recall values. Notably, NILS demonstrates stable performance across varying node degrees, with only a marginal decrease in effectiveness for high-degree target nodes. This can be attributed to the smaller influence that each neighboring node has on the aggregation of the GCN layer when the target node degree is high.

Table 5.3 compares NILS with LinkTeller and the link-stealing attacks (LSA2-post and LSA2-attr) introduced in [54] on the citation network datasets. These attacks are executed under the transductive setting, where training and inference occur on the same graph. The results show that NILS outperforms the LSA2-post and LSA2-attr attacks across all three datasets. However, NILS's performance is nearly equivalent to that of LinkTeller in this transductive setting. These results demonstrate that NILS maintains effectiveness under the transductive setting, just as in the inductive setting, while consistently outperforming other baseline attacks.

5.4.3 Impact of GNN Depth

We examined the impact of increasing the depth of GNN on the success rate of the NILS attack for the Twitch-FR dataset. Figure 5.2 illustrates

Dataset	Method	low		unconstrained		high	
		precision	recall	precision	recall	precision	recall
TWITCH-FR	NILS (Ours)	100.0 $\pm$0.0	100.0 $\pm$0.0	99.13 $\pm$0.8	99.57 $\pm$0.35	99.91 $\pm$2.6	100.0 $\pm$0.0
	LinkTeller	92.5 $\pm$5.4	92.5 $\pm$5.4	84.1 $\pm$3.7	78.2 $\pm$1.9	83.2 $\pm$1.4	80.6 $\pm$6.7
TWITCH-RU	NILS (Ours)	100.0 $\pm$0.0	100.0 $\pm$0.0	96.45 $\pm$0.4	98.34 $\pm$0.7	99.77 $\pm$0.1	99.37 $\pm$0.1
	LinkTeller	78.8 $\pm$1.9	92.6 $\pm$5.5	71.8 $\pm$2.2	78.5 $\pm$2.4	89.7 $\pm$1.7	65.7 $\pm$3.9
Flickr	NILS (Ours)	100.0 $\pm$0.0	100.0 $\pm$0.0	99.11 $\pm$1.7	95.83 $\pm$5.0	93.72 $\pm$3.1	78.9 $\pm$1.9
	LinkTeller	51.0 $\pm$7.0	53.3 $\pm$4.7	33.8 $\pm$13.3	32.1 $\pm$13.3	18.2 $\pm$4.5	18.5 $\pm$6.1

Table 5.2: Comparative performance of NILS and LinkTeller across three datasets (TWITCH-FR, TWITCH-RU, and Flickr) under low, unconstrained, and high constraint settings. The results are presented in terms of precision and recall with corresponding standard deviations.

Method	Cora		Citeseer		Pubmed	
	precision	recall	precision	recall	precision	recall
NILS (Ours)	99.7 $\pm$0.2	99.6 $\pm$0.3	97.4 $\pm$0.2	98.2 $\pm$0.1	99.7 $\pm$0.0	100.0 $\pm$0.0
LinkTeller	99.5 $\pm$0.1	99.5 $\pm$0.1	99.7 $\pm$0.0	99.7 $\pm$0.0	99.7 $\pm$0.0	99.7 $\pm$0.0
LSA2-post	86.7 $\pm$0.2	86.7 $\pm$0.2	90.1 $\pm$0.2	90.1 $\pm$0.2	78.8 $\pm$0.1	78.8 $\pm$0.1
LSA2-attr	73.6 $\pm$0.1	73.6 $\pm$0.1	80.9 $\pm$0.1	80.9 $\pm$0.1	82.4 $\pm$0.1	82.4 $\pm$0.1

Table 5.3: Comparative performance of NILS with LinkTeller [138] and link-stealing attacks in [54] across three datasets (Cora, Citeseer, and Pubmed).

the attack's success rates for different GNN depths and malicious feature generation strategies.

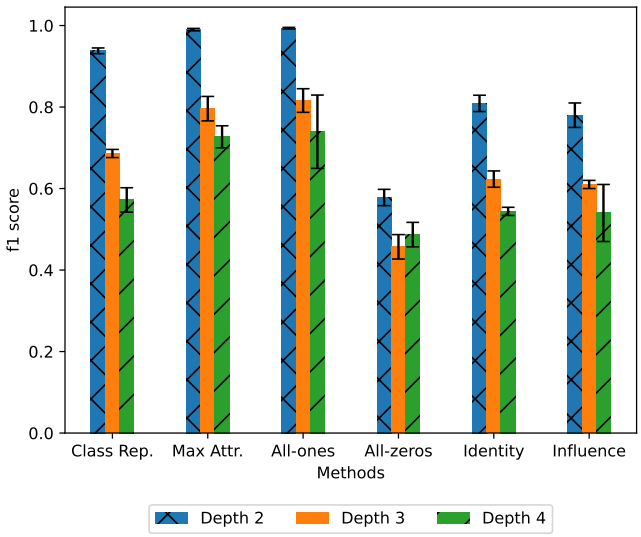


Figure 5.2: Success rates of the NILS attack for different depths and malicious features generation strategies for the Twitch-FR dataset.

Our findings indicate that as the depth of the GNN increases, the attack's success rate decreases. This can be attributed to the dilution of the injected poisoning node's influence within the target node's neighborhood. As the GNN depth increases, the model aggregates information from a larger neighborhood, encompassing nodes that are k-hops away from the target node. Consequently, the injected malicious node's features become one among many contributing factors in the aggregated information, leading to a dilution of its influence. This reduction in the injected node's impact on the aggregated information diminishes the overall effectiveness of the attack, making it less successful in altering the predictions of the target node's neighbors.

Table 5.4 compares the performance of NILS and LinkTeller across different GCN depths for the Twitch-FR and Twitch-RU datasets.

Dataset	Method	Depth-2		Depth-3	
		precision	recall	precision	recall
TWITCH-FR	NILS (Ours)	99.13 $\pm$0.8	99.57 $\pm$0.35	85.06 $\pm$1.2	81.56 $\pm$1.2
	LinkTeller	84.1 $\pm$3.7	78.2 $\pm$1.9	50.1 $\pm$5.1	46.6 $\pm$5.0
TWITCH-RU	NILS (Ours)	96.45 $\pm$0.4	98.34 $\pm$0.7	78.78 $\pm$3.8	76.35 $\pm$9.3
	LinkTeller	71.8 $\pm$2.2	78.5 $\pm$2.4	45.7 $\pm$2.2	50.0 $\pm$2.8

Table 5.4: Success rates of the attack for different depths in comparison with LinkTeller [138]. We use the all-ones strategy for NILS.

The results in Table 5.4 demonstrate that NILS outperforms LinkTeller across various GCN depths. For the Twitch-FR dataset, NILS shows higher precision and recall values when the GCN depth is 3 (precision: 85.06 $\pm$1.2, recall: 81.56 $\pm$1.2) compared to the LinkTeller method (precision: 50.01 $\pm$5.1, recall: 46.6 $\pm$5.0). Notably, NILS consistently outperforms LinkTeller even when comparing the attack performance of LinkTeller with a GCN depth of 2 and NILS with a GCN depth of 3. These results highlight the effectiveness of our node injection strategy, as it consistently outperforms the LinkTeller method across different depths of the GCN.

5.4.4 Discussion

The experimental results demonstrate the effectiveness of the NILS attack across various datasets and settings. The attack's success can be attributed to several factors:

1. **Exploitation of GNN dynamics**: By injecting a new node and connecting it to the target node, NILS takes advantage of the dynamic nature of GNNs. This approach allows for a more direct influence on the target node's neighborhood, leading to more significant changes in prediction scores.

2. **Effective feature generation strategies**: The Max attributes and All-ones strategies consistently perform well across different datasets. These strategies create feature vectors that maximize the impact on the GNN's predictions, making it easier to distinguish between neighbors and non-neighbors of the target node.

3. **Robustness across node degrees**: NILS demonstrates stable performance across varying node degrees, with only a slight decrease in effectiveness for high-degree target nodes. This robustness is a significant advantage over previous attacks.

4. **Performance in both inductive and transductive settings**: The attack maintains its effectiveness in both inductive (Twitch and Flickr datasets) and transductive (citation network datasets) settings, showcasing its versatility.

However, the attack's performance does decrease as the depth of the GNN increases. This limitation is due to the dilution of the injected node's influence in deeper networks, as information from a larger neighborhood is aggregated. This observation suggests that deeper GNN architectures may offer some inherent resistance to this type of attack, albeit at the potential cost of increased computational complexity and the risk of over-smoothing.

The comparison with baseline attacks, particularly LinkTeller and the link-stealing attacks in [54], highlights the superiority of NILS in most scenarios. The significant improvements in F1 scores, especially on the Flickr dataset, demonstrate the power of the node injection approach combined with carefully crafted malicious features.

These results have important implications for the privacy and security of GNN-based systems. They underscore the need for robust defense mechanisms that can protect against sophisticated attacks like NILS, which exploit the dynamic nature of these models. In the next section, we will explore potential defense strategies based on differential privacy, specifically tailored to counter the NILS attack.

5.5 Defense Strategy

As demonstrated in the previous section, the NILS attack poses a significant threat to the privacy of graph-structured data in GNNs. This section introduces a defense strategy based on differential privacy (DP) to counter the NILS attack. We first present a new DP notion tailored to our attack scenario, then describe the LapGraph mechanism adapted to satisfy this notion, and finally evaluate its effectiveness in mitigating the NILS attack.

5.5.1 One-Node-One-Edge-Level Differential Privacy

The NILS attack, as described in Section 5.2.3, involves adding a malicious node to a graph and connecting it to a target node through a single edge. To

counter such an adversary, we introduce a new notion of differential privacy specifically designed for this scenario.

Definition 14 (One-node-one-edge-level adjacent graphs)**.** $\mathcal{G}$ and $\mathcal{G}'$ are considered *one-node-one-edge-level adjacent graphs* if one can be obtained from the other by adding a single node with one edge only.

Based on this definition of graph adjacency, we define one-node-one-edge-level DP as follows:

Definition 15 ((ε, δ)-One-node-one-edge-level DP)**.** A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-one-node-one-edge-level DP with $\varepsilon, \delta \geqslant 0$ if, for all pairs of one-node-one-edge-level adjacent graphs $\mathcal{G}, \mathcal{G}'$ and for all measurable $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$, the following holds:

$$\text{P}\{\mathcal{M}(\mathcal{G}) \in \mathcal{O}\} \leqslant e^{\varepsilon}\, \text{P}\{\mathcal{M}(\mathcal{G}') \in \mathcal{O}\} + \delta$$

This definition ensures that the output of a mechanism satisfying one-node-one-edge-level DP is similarly distributed for any pair of graphs that differ by at most one node and one edge, up to a factor of e^{ε} and an additive term δ.

5.5.2 LapGraph Mechanism for One-Node-One-Edge-Level DP

To achieve one-node-one-edge-level DP, we adapt the LapGraph mechanism introduced in [138]. The LapGraph mechanism consists of perturbing the adjacency matrix using the Laplace mechanism and then binarizing it by replacing the top-N largest values with 1 and the remaining values with 0, where N is the estimated number of edges in the graph.

Let f_A be the query function returning the adjacency matrix of a graph G. For two one-node-one-edge neighboring graphs G, G', their corresponding matrices A, A' have different dimensions: $A \in \mathbb{R}^{n \times n}$ and $A' \in \mathbb{R}^{(n+1)\times(n+1)}$, or vice versa. To enable the computation of the sensitivity of f_A, we append one zero-row and one-zero column to the smaller matrix, resulting in $\bar{A} \in \mathbb{R}^{(n+1)\times(n+1)}$.

The L_1-global sensitivity of f_A for one-node-one-edge neighboring graphs is:

$$\Delta_1(f_A) = \max_{\substack{\forall G, G' \\ \text{1N1E-adjacent}}} \|\bar{A} - A'\|_1 = 1$$

This sensitivity is the same as in the original LapGraph mechanism intended for edge-level DP. Consequently, the LapGraph mechanism can be directly applied to achieve one-node-one-edge-level DP, providing stronger protection for the same level of utility compared to edge-level DP.

The LapGraph mechanism for one-node-one-edge-level DP can be described as follows:

1. Add Laplace noise to each element of the adjacency matrix:

   $$\tilde{A}_{ij} = A_{ij} + \text{Lap}(1/\varepsilon)$$

   where $\text{Lap}(1/\varepsilon)$ denotes a random variable drawn from the Laplace distribution with scale $1/\varepsilon$.

2. Estimate the number of edges N using the Laplace mechanism:

   $$\tilde{N} = N + \text{Lap}(1/\varepsilon)$$

3. Binarize the noisy adjacency matrix by setting the top $\tilde{N}$ elements to 1 and the rest to 0.

5.5.3 Evaluation of LapGraph Defense

We evaluated the effectiveness of the LapGraph mechanism in reducing the success of the NILS attack while ensuring one-node-one-edge-level DP. We also investigated the utility of GCN models trained with LapGraph protection.

Experimental Setup

We used the same training hyperparameters and normalization techniques as in the non-private case. The training graph was initially protected with LapGraph, and the mechanism was reapplied each time the graph changed due to node injection by the adversary. We computed the F_1 score for the NILS attack and the classification task's F_1 score for the GCN across various privacy budgets ε. Results were averaged over 5 runs with different random seeds for LapGraph.

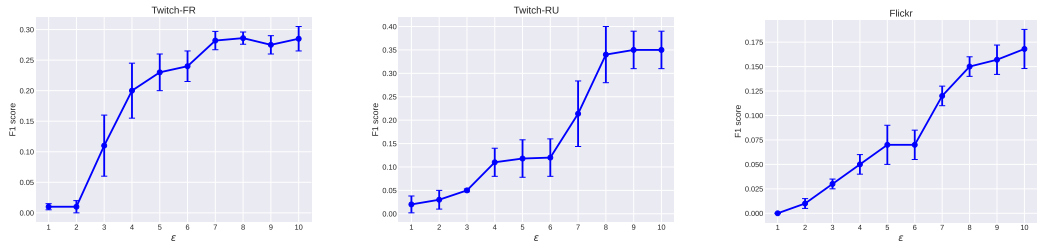


Figure 5.3: F_1 score of the NILS attack for different values of ε.

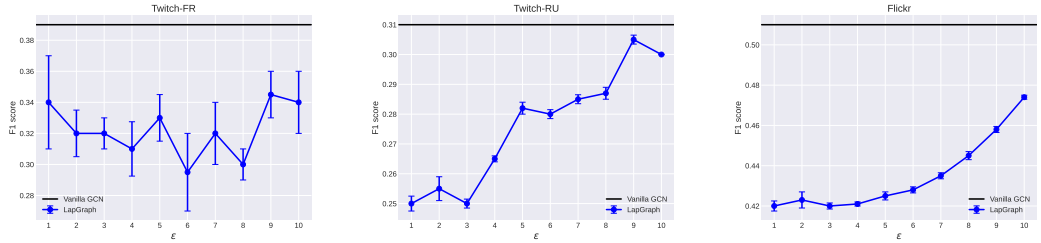


Figure 5.4: F_1 score utility of the GCN for different values of ε.

Results and Analysis

Figure 5.3 presents the F_1 score of the NILS attack for various ε values across the Twitch-FR, Twitch-RU, and Flickr datasets.

The results show that applying LapGraph significantly reduces the effectiveness of NILS. For small privacy budgets ($\varepsilon < 1$), the attack's F_1 score approaches zero across all datasets. As ε increases, the attack's effectiveness gradually improves, but remains substantially lower than in the non-private case.

This defense proves more robust compared to its application against the LinkTeller attack [138], where LapGraph offered limited protection for large ε values. The enhanced protection in our scenario can be attributed to the application of LapGraph not only during training but also after each node injection, making it more challenging for the adversary to distinguish between the target node's neighbors and non-neighbors.

To assess the privacy-utility trade-off of LapGraph, we evaluated the utility of the GCNs for different privacy budgets. Figure 5.4 presents these results.

As expected, the utility increases with larger ε values. For $\varepsilon \geq 7$, the utility approaches that of the non-private case. This suggests that careful selection of ε can provide a balance between maintaining good utility and offering protection against the NILS attack.

5.5.4 Discussion

The proposed LapGraph mechanism, adapted to satisfy one-node-one-edge-level DP, demonstrates significant effectiveness in mitigating the NILS attack. By applying noise to the adjacency matrix and reapplying the mechanism after each node injection, the defense makes it challenging for the adversary to infer the true structure of the graph.

However, this protection comes at the cost of reduced utility, especially for smaller privacy budgets. The privacy-utility trade-off observed in our experiments aligns with the fundamental challenge in differential privacy: stronger privacy guarantees typically result in lower utility.

Compared to the DP-based defenses discussed for PCA in Chapter 4, the LapGraph mechanism offers a more tailored approach for graph-structured data. While both approaches add noise to protect privacy, LapGraph's binarization step helps maintain the graph structure, which is crucial for GNN performance. This difference highlights how privacy-preserving mechanisms must be carefully designed to match the specific characteristics of the data and models they aim to protect.

The effectiveness of this defense strategy underscores the importance of considering the specific attack model when designing privacy-preserving mechanisms. By tailoring our DP notion to the NILS attack, we achieved stronger protection compared to generic edge-level DP, without incurring additional utility costs.

5.6 Conclusion

In this chapter, we have presented a powerful new NILS attack—a link-stealing attack using node injection against GNNs. Our results have demonstrated the superior performance of NILS compared to previous attacks, further emphasizing the vulnerabilities of GNNs regarding edge information leakage. We have also evaluated NILS against differentially private GNNs, ensuring a one-node-one-edge-level DP notion specifically designed to protect against our proposed attack. The NILS attack significantly enhances the F_1 score of the attack beyond the current state-of-the-art benchmarks. Specifically, for the Twitch dataset, our method improves the F_1 score by 23.75%, and for the Flickr dataset, it records a remarkable improvement, where the new performance is more than three times better than the state-of-the-art. We also proposed and evaluated defense strategies based on differentially private (DP) mechanisms relying on a newly defined DP notion. Our LapGraph-based defense, on average, reduces the effectiveness of the

attack by approximately 71.9% while only incurring a minimal average utility loss of about 3.2%. These findings underscore the ongoing challenges in preserving privacy in graph-structured data and the need for robust defense mechanisms against sophisticated attacks. As we continue to explore the landscape of privacy in machine learning, the next chapter will build upon these insights, extending our investigation to the realm of federated learning. Next Chapter will examine link stealing attacks in the context of vertical federated graph learning, exploring how the distributed nature of federated learning introduces new privacy challenges in graph-based models.

Chapter 6

Link Stealing Attacks in Vertical Federated Graph Learning

6.1 Introduction

The advent of graph data in artificial intelligence has marked a significant milestone in enhancing the performance and accuracy of machine learning models. This surge in graph data usage, while promising, has introduced new challenges in terms of privacy and security. Recent studies have unveiled vulnerabilities in Graph Neural Networks (GNN), particularly their susceptibility to link inference attacks (LIA) [54]. These attacks aim to uncover relationships among graph nodes by identifying or inferring the existence of edges between them, potentially exposing sensitive information about the relationships or interactions between parties represented by nodes in the graph.

In parallel with the rise of graph-based learning, federated learning has emerged as a pivotal paradigm in collaborative machine learning. Introduced in [87], federated learning enables multiple parties to collaboratively train machine learning models while keeping their data on local premises, thereby ensuring data privacy. The integration of graph data with federated learning has given rise to federated graph learning, which can be implemented in two settings: horizontal and vertical. In the horizontal setting, multiple parties collaborate to train a global model using their local graph datasets that share similar feature spaces but differ in samples. In contrast, the vertical federated graph learning (VFGL) setting involves each collaborating party holding different features of the same samples.

VFGL presents a unique scenario where one party may possess a graph dataset, while another may have features about the samples without any associated graph topology. In this setting, parties utilize their local datasets,

which may include graphs, to train local models that generate intermediate or latent representations of their data. These representations are then sent to a server, which combines them along with its own training labels to train its model. This approach allows for leveraging both graph structures and feature data without sharing raw data, thus preserving user privacy while enhancing model effectiveness.

However, the VFGL architecture introduces potential vulnerabilities where clients participating in the training of the model, but not holding the actual graph data, can become adversaries and infer information about the edges/links in the graph from the available training information. This information can range from the gradients shared among parties during each training epoch to the model output when queried. The nature of these attacks varies based on the party that can perform them, namely any participating client or the server itself.

While some attacks relying on the knowledge of classification output or intermediate results have been proposed and evaluated [104], the potential for link inference attacks based on gradients or training labels remains unexplored in the context of VFGL. This gap in knowledge presents a critical area for investigation, given the central role of gradients in the training process and the potential for label information to be embedded within them.

In this chapter, we address this gap by introducing and studying two novel link inference attacks in the VFGL setting: a gradient-based LIA and a label-based LIA. Our research aims to:

1. Construct and evaluate a new LIA that exploits the gradients information during training.

2. Investigate the effectiveness of our proposed attacks using seven real-world datasets, comparing their performance against existing forms of LIA.

3. Identify the underlying reasons for the success of our proposed Gradient-based LIA, particularly its relation to node label information embedded in the gradients.

4. Develop and analyze a new Label-based LIA to demonstrate its strong connection to our primarily proposed Gradient-based LIA.

5. Provide a theoretical study on how the success of label-based LIA is influenced by graph properties such as homophily, density, and class diversity.

6. Evaluate defense mechanisms against our proposed attacks, including edge-level perturbation and a novel label perturbation approach, demonstrating their effectiveness in mitigating the attacks while maintaining a balance between privacy and utility.

Our findings reveal that gradients in VFGL can indeed leak significant link information, primarily due to the embedded label information within the gradients. We demonstrate that our proposed attacks often outperform other forms of LIA, including those based on model output predictions, which typically assume a stronger adversary. Furthermore, we provide analytical insights into why these attacks are effective and identify potential defenses, highlighting the need for further research to improve the security of VFGL systems against link information leakage.

The remainder of this chapter is organized as follows: Section 6.2 details our proposed link inference attacks and adversary models. Section 6.3 presents the analytical results for link inference attacks. Section 6.4 details our experimental setup. In Section 6.5, we present our experimental results, followed by a comprehensive discussion of our findings. Section 6.6 introduces and evaluates defense mechanisms against the proposed attacks. Finally, Section 6.7 concludes the chapter with a summary of our contributions and directions for future research.

6.2 Attack Methodology

This section presents a detailed description of the link inference attacks (LIA) developed in this study. We first introduce the Vertical Federated Learning (VFL) system that forms the basis of our attack scenarios, followed by the description of our novel attacks: Gradient-based LIA and Label-based LIA, and their comparison with existing baseline attacks.

6.2.1 VFL system

In this study, we investigate a two-party VFL setting involving parties $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$, along with an active party $\mathcal{P}_\mathcal{Y}$. $\mathcal{P}_\mathcal{G}$ owns a graph dataset denoted as $\mathcal{D}_\mathcal{G} = (\mathcal{G}, X_\mathcal{G})$, while $\mathcal{P}_\mathcal{A}$ holds a separate feature set denoted as $X_\mathcal{A}$. The parties share a user space of N samples, implying that the graph $\mathcal{G}$ contains N nodes, each representing a user. Within this user space, $\mathcal{P}_\mathcal{G}$ and $\mathcal{P}_\mathcal{A}$ manage user features of dimensions d_G and d_A respectively. They collaborate with $\mathcal{P}_\mathcal{Y}$, the party owning the classification labels $\mathcal{Y}$, to perform a supervised learning task.

Table 6.1: Table of Notations

Notation	Description
$\mathcal{P}_{\mathcal{G}}$	Party owning the graph dataset
$\mathcal{P}_{\mathcal{A}}$	Party holding the separate feature set
$\mathcal{P}_{\mathcal{Y}}$	Party owning the training labels
$\mathcal{G}$	Target graph owned by $\mathcal{P}_{\mathcal{G}}$
$X_{\mathcal{G}}$	Features owned by party $\mathcal{P}_{\mathcal{G}}$
$H_{\mathcal{G}}$	Intermediate representation of $X_{\mathcal{G}}$
$G_{\mathcal{G}}$	The gradients sent to party $P_{\mathcal{G}}$
$X_{\mathcal{A}}$	Features owned by party $\mathcal{P}_{\mathcal{A}}$
$H_{\mathcal{A}}$	Intermediate representation of $X_{\mathcal{A}}$
P	Output prediction computed by $\mathcal{P}_{\mathcal{Y}}$
$G_{\mathcal{A}}$	The gradients sent to party $P_{\mathcal{A}}$
$\mathcal{Y}$	Training labels owned by party $\mathcal{P}_{\mathcal{Y}}$

Figure 6.1 illustrates the VFL setting with two clients and one server. Specifically, $\mathcal{P}_{\mathcal{G}}$ employs a Graph Neural Network (GNN) to transform $X_{\mathcal{G}}$ into its intermediate representation $H_{\mathcal{G}}$, while $\mathcal{P}_{\mathcal{A}}$ uses a deep neural network to transform $X_{\mathcal{A}}$ into its intermediate representation $H_{\mathcal{A}}$. $\mathcal{P}_{\mathcal{Y}}$ gathers these intermediate representations and trains a Deep Neural Network (DNN) to compute the output prediction P for classification.

The process of training involves computing the loss function $\mathcal{L}$, deriving the gradients with respect to the model parameters, and then updating these parameters. The gradients are computed according to the following rule:

$$\nabla_{\theta_k}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_i \frac{\partial \mathcal{L}}{\partial H_{i,k}} \frac{\partial H_{i,k}}{\partial \theta_k} \tag{6.1}$$

where θ_k represents the model parameters, $H_{i,k}$ is the latent representation of the i^{th} sample computed by party $\mathcal{P}_k$, and $\frac{\partial \mathcal{L}}{\partial H_{i,k}}$ is the gradient of the loss function $\mathcal{L}$ with respect to $H_{i,k}$ for $k \in \{\mathcal{G}, \mathcal{A}\}$. The details of the VFL training protocol are outlined in Algorithm 4.

In this VFL setting, $\mathcal{P}_{\mathcal{G}}$ is considered the victim party as it holds the graph dataset, while $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{Y}}$ can potentially launch different Link Inference Attacks (LIA). We now describe these attacks in detail.

6.2.2 Gradient-based LIA

The Gradient-based LIA leverages the gradients received by a participating client during the VFGL training process. In this attack, the adversary ex-

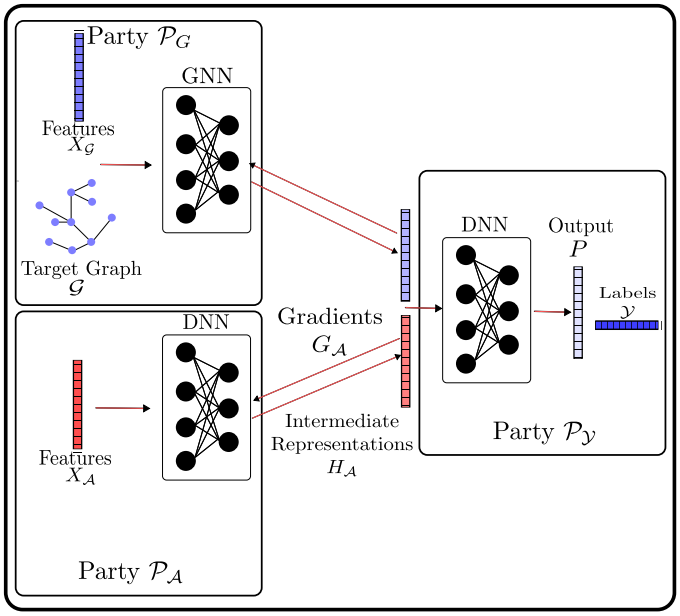


Figure 6.1: Schematic representation of a VFL setting with two clients and one server

ploits the gradients $G_{\mathcal{A}}$ of node samples to infer links in the target graph $\mathcal{G}$. The attack proceeds as follows:

1. The adversary receives gradients $G_{\mathcal{A}}$ from the server for the cut layer of their model.

2. For each pair of samples (i, j), the adversary computes the cosine similarity between their gradients:

$$S = \text{Cosine similarity}(G_{\mathcal{A}}^{(i)}, G_{\mathcal{A}}^{(j)}) \tag{6.2}$$

3. The similarity score is compared against a threshold value τ. If $S > \tau$, the adversary infers a link between samples i and j in the reconstructed graph $\hat{\mathcal{G}}$.

The choice of cosine similarity as the metric is motivated by its effectiveness in measuring sample similarity from a neural network perspective [10, Corollary 2] and its common usage in gradient-based FL attacks [42]. Empirically, cosine similarity outperforms other distance metrics such as Euclidean and Chebyshev distances in our attack scenario, as it captures the angle between vectors rather than magnitude differences.

Algorithm 5 provides a detailed description of the Gradient-based LIA.

Algorithm 4 Two-Party Vertical Federated Learning

Require: learning rates $\eta_{\mathcal{G}}$ and $\eta_{\mathcal{A}}$
Ensure: Trained model parameters $\theta_{\mathcal{G}}$, $\theta_{\mathcal{A}}$, ψ
Parties $\mathcal{P}_{\mathcal{G}}$, $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{Y}}$ initialize $\theta_{\mathcal{G}}$, $\theta_{\mathcal{A}}$, ψ.
for each training iteration $t = 1, 2, \ldots$ **do**
    In **parallel** do the following:
        Party $\mathcal{P}_{\mathcal{G}}$:
            Computes $H_{\mathcal{G}} = GNN(X_{\mathcal{G}}, \theta_{\mathcal{G}})$
            Sends $H_{\mathcal{G}}$ to party $\mathcal{P}_{\mathcal{Y}}$
        Party $\mathcal{P}_{\mathcal{A}}$:
            Computes $H_{\mathcal{A}} = DNN(X_{\mathcal{A}}, \theta_{\mathcal{A}})$
            Sends $H_{\mathcal{A}}$ to party $\mathcal{P}_{\mathcal{Y}}$
    End
    Party $\mathcal{P}_{\mathcal{Y}}$ computes the prediction output $P_{\mathcal{Y}} = DNN((X_{\mathcal{G}}, X_{\mathcal{A}}), \psi)$
    $\mathcal{P}_{\mathcal{Y}}$ updates $\psi^{t+1} = \psi^t - \eta_{\mathcal{G}} \frac{\partial \mathcal{L}}{\partial \psi}$
    $\mathcal{P}_{\mathcal{Y}}$ computes and sends the gradients $G_{\mathcal{G}} = \frac{\partial \mathcal{L}}{\partial H_{\mathcal{G}}}$ and $G_{\mathcal{A}} = \frac{\partial \mathcal{L}}{\partial H_{\mathcal{A}}}$ to $\mathcal{P}_{\mathcal{G}}$ and $\mathcal{P}_{\mathcal{A}}$, respectively.
    In **parallel** do the following:
        Party $\mathcal{P}_{\mathcal{G}}$:
            Computes $\nabla_{\theta_{\mathcal{G}}} \mathcal{L}$ with Equation 6.1
            Updates $\theta_{\mathcal{G}}^{t+1} = \theta_{\mathcal{G}}^t - \eta_{\mathcal{G}} \nabla_{\theta_{\mathcal{G}}} \mathcal{L}$
        Party $\mathcal{P}_{\mathcal{A}}$:
            Computes $\nabla_{\theta_{\mathcal{A}}} \mathcal{L}$ with Equation 6.1
            Updates $\theta_{\mathcal{A}}^{t+1} = \theta_{\mathcal{A}}^t - \eta_{\mathcal{A}} \nabla_{\theta_{\mathcal{A}}} \mathcal{L}$
    End
end for

Selecting an optimal threshold τ is challenging without prior knowledge of the target graph's structure. We propose several practical strategies for threshold selection:

- If the adversary can estimate the graph's density d, they could select the top $d\%$ most similar node pairs as connected, using the least similar among these as the threshold.
- Alternatively, the adversary might use a public partial graph or a similar graph in the same domain to estimate an appropriate threshold.

6.2.3 Label-based LIA

The Label-based LIA exploits the training labels $\mathcal{Y}$ of the samples to infer links in the target graph $\mathcal{G}$. This attack assumes that nodes with the same

Algorithm 5 Gradient-based LIA

Require: Gradients $G_{\mathcal{A}}$ and threshold τ
Ensure: Inferred graph $\hat{\mathcal{G}}$
1: Adversary $\mathcal{A}$ receives gradients $G_{\mathcal{A}}$ from the server.
2: $\mathcal{A}$ initializes an empty graph $\hat{\mathcal{G}}$.
3: **for** each pair of samples (i, j) **do**
4: $S \leftarrow$ Cosine similarity$(G_{\mathcal{A}}^{(i)}, G_{\mathcal{A}}^{(j)})$.
5: **if** $S > \tau$ **then**
6: $\mathcal{A}$ infers a link between samples i and j in graph $\hat{\mathcal{G}}$.
7: **end if**
8: **end for**

labels are more likely to be connected. The attack proceeds as follows:

1. For each pair of samples (i, j), the adversary compares their labels $\mathcal{Y}_i$ and $\mathcal{Y}_j$.

2. If $\mathcal{Y}_i = \mathcal{Y}_j$, the adversary infers a link between samples i and j in the reconstructed graph $\hat{\mathcal{G}}$.

Algorithm 6 provides a detailed description of the Label-based LIA.

Algorithm 6 Label-based LIA

Require: Labels $\mathcal{Y}$
Ensure: Inferred graph $\hat{\mathcal{G}}$
1: $\mathcal{A}$ initializes an empty graph $\hat{\mathcal{G}}$.
2: **for** each pair of samples (i, j) **do**
3: **if** $\mathcal{Y}_i == \mathcal{Y}_j$ **then**
4: $\mathcal{A}$ infers a link between samples i and j in graph $\hat{\mathcal{G}}$.
5: **end if**
6: **end for**

It is important to note that the Label-based LIA requires different knowledge compared to the Gradient-based LIA, specifically access to training labels. While this assumption might seem strong, it is not unrealistic in certain scenarios. For instance, in our VFGL protocol, the server has access to the labels to perform the final model training. The Label-based LIA was introduced to provide deeper insights into the Gradient-based LIA's effectiveness, highlighting the complementary nature of the two methods in understanding link inference attacks in VFGL.

6.2.4 Baseline LIA

To provide a comprehensive evaluation of our proposed attacks, we compare them with several baseline LIA from existing literature. These baseline attacks follow a similar principle to the Gradient-based LIA but utilize different observations of samples. The baseline LIA we consider are:

- Intermediate Representation-based LIA: This attack uses the intermediate representations of samples, as studied in [104].
- Output-based LIA: This attack leverages the output predictions of the model, also explored in [104].
- Feature-based LIA: This attack utilizes the features of samples, as investigated in [54].

The general procedure for these baseline attacks is as follows:

1. The adversary collects the relevant observations ($\mathcal{O}$) of samples based on the specific attack type.
2. For each pair of samples (i, j), the adversary computes the cosine similarity between their observations:

$$S = \text{Cosine similarity}(\mathcal{O}_i, \mathcal{O}_j) \tag{6.3}$$

3. The similarity score is compared against a threshold value τ. If $S > \tau$, the adversary infers a link between samples i and j in the reconstructed graph $\hat{\mathcal{G}}$.

Algorithm 7 provides a unified description of these baseline LIA.

These baseline attacks provide a comprehensive set of comparisons for our proposed Gradient-based and Label-based LIA, allowing us to evaluate the effectiveness of our attacks in the context of existing methods.

To evaluate the effectiveness of our attacks comprehensively, we primarily use the Area Under the Curve (AUC) metric, following previous works [149, 104, 110, 54]. This metric assesses performance across various thresholds, providing a threshold-independent evaluation. For comparison with the Label-based attack, which doesn't rely on a threshold, we compute the accuracy of our threshold-dependent attacks at the threshold yielding the highest F1 score. This approach balances precision and recall, offering detailed insights into each attack strategy's effectiveness while accounting for the practical challenges of threshold selection in real-world scenarios.

Algorithm 7 Baseline LIA

Require: Observations ($\mathcal{O}$) of samples based on attack type (intermediate representations, outputs, or features) and threshold τ
Ensure: Inferred graph $\hat{\mathcal{G}}$

Adversary $\mathcal{A}$ collects observations depending on the specific attack focus.
$\mathcal{A}$ initializes an empty graph $\hat{\mathcal{G}}$.
for each pair of samples (i, j) **do**
    $S \leftarrow$ Cosine similarity$(\mathcal{O}_i, \mathcal{O}_j)$.
    if $S > \tau$ **then**
        $\mathcal{A}$ infers a link between samples i and j in graph $\hat{\mathcal{G}}$.
    end if
end for

It's worth noting that while the Gradient-based and Intermediate representation-based LIA are initially designed for participating clients, the server also has the necessary knowledge to perform these attacks. This is because the server dispatches gradients to the clients and receives the intermediate representations of the nodes from them. On the other hand, the Label-based LIA and Output-based LIA can only be executed by the server, as it has exclusive access to the training labels and final model outputs.

In the context of our VFL setting, $\mathcal{P}_{\mathcal{G}}$ is considered the victim party as it holds the graph dataset. Both $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{Y}}$ can potentially act as adversaries, launching different types of LIA based on the information available to them. The Gradient-based, Intermediate representation-based, and Feature-based LIA can be executed by $\mathcal{P}_{\mathcal{A}}$, while $\mathcal{P}_{\mathcal{Y}}$ can perform all types of attacks, including the Label-based and Output-based LIA.

The introduction of the Gradient-based LIA represents a significant contribution of this work. While intermediate representations and prediction outputs were already exploited in [104] and features in [54], the potential of gradient information for link inference in the VFGL setting had not been previously studied. Our work demonstrates that gradients can be a powerful source of information for inferring graph structure, potentially outperforming existing methods in certain scenarios.

The Label-based LIA, while simpler in nature, serves an important purpose in our study. It provides a baseline for understanding the relationship between node labels and graph structure, and helps to elucidate why the Gradient-based LIA is effective. As we will show in the following sections, the performance of the Gradient-based LIA is closely related to that of the Label-based LIA, suggesting that the gradients carry significant label information.

To provide a clear overview of the different attacks and the knowledge required by the adversary to perform them, we present Table 6.2.

Table 6.2: Adversary capabilities in different attack scenarios

Party	Features	Labels	Gradients	Inter-Rep.	Pred. Output
Client	✓	✗	✓	✓	✗
Server	✗	✓	✓	✓	✓

Table 6.2 summarizes the information available to different parties in the VFL setting. The client ($\mathcal{P}_{\mathcal{A}}$) has access to features, gradients, and intermediate representations, allowing it to perform Gradient-based, Intermediate representation-based, and Feature-based LIA. The server ($\mathcal{P}_{\mathcal{Y}}$), on the other hand, has access to labels, gradients, intermediate representations, and prediction outputs, enabling it to perform all types of attacks including the Label-based and Output-based LIA.

This table highlights the different attack vectors available to each party and underscores the importance of considering multiple potential adversaries when designing privacy-preserving VFL systems. It also illustrates why our proposed Gradient-based LIA is particularly significant, as it can be executed by both the client and the server, potentially exposing graph structure information to multiple parties in the VFL setting.

In the next section, we will provide analytical results for these LIA, focusing on the theoretical performance guarantees of the Label-based LIA and its relationship with the Output-based LIA. We will also present a toy example illustrating the connection between the Gradient-based LIA and the Label-based LIA, providing insights into the effectiveness of our proposed attacks. These analyses will help to deepen our understanding of the vulnerabilities in VFGL systems and guide the development of more robust privacy-preserving techniques.

6.3 Analytical Results for Link Inference Attacks

In this section, we provide theoretical guarantees for the performance of our proposed Link Inference Attacks (LIA). We first analyze the Label-based LIA, deriving its accuracy based on graph properties. Then, we demonstrate a special case where the Output-based LIA achieves the same link inferences as the Label-based LIA. Finally, we present a toy example illustrating the close relationship between the Gradient-based LIA and the Label-based LIA.

Performance of Label-based LIA

The Label-based Link Inference Attack (LIA) operates on the principle of inferring a link between two nodes when they share the same label. The effectiveness of this attack is closely tied to the correlation between links and node labels within the graph structure. To quantify this correlation and analyze the attack's performance, we introduce two key concepts: the homophily ratio and class diversity.

Definition 16 (Homophily Ratio [161])**.** The *homophily ratio* of a graph quantifies the likelihood that adjacent nodes in the graph share the same label. Formally, the homophily ratio h can be expressed as:

$$h = \frac{|\{(v, w) : (v, w) \in E \wedge \mathcal{Y}_v = \mathcal{Y}_w\}|}{|E|}, \tag{6.4}$$

where E denotes the set of edges in the graph, v and w represent nodes, and $\mathcal{Y}_v$ and $\mathcal{Y}_w$ are their respective class labels.

While the homophily ratio is crucial for understanding the success of inference on existing links, it does not fully capture the attack's performance on non-existing links. To address this, we introduce a class diversity metric:

Definition 17 (Graph Density)**.** The density d of a graph $G = (V, E)$ is defined as the ratio of the number of edges in the graph to the number of edges in a complete graph with the same number of vertices. For an undirected graph, it is given by:

$$d = \frac{2|E|}{|V|(|V| - 1)} \tag{6.5}$$

where $|E|$ is the number of edges and $|V|$ is the number of vertices in the graph. The density ranges from 0 for a graph with no edges to 1 for a complete graph.

Definition 18 (Class Diversity)**.** The *class diversity* of a graph with C distinct label classes is defined as:

$$1 - \sum_{c=1}^{C} \alpha_c^2 \tag{6.6}$$

where α_c represents the proportion of nodes with label c.

With these definitions in place, we can now state the theorem on the accuracy of the Label-based LIA:

Theorem 6 (Accuracy of Label-based Link Inference Attack)**.** For a graph G with N nodes, exhibiting homophily ratio h, density d, and C distinct label classes, let $\alpha_c = \frac{N_c}{N}$ represent the proportion of nodes with label c. The accuracy Acc of the Label-based link inference attack can be computed as:

$$Acc = 2hd - d + \frac{N}{N-1}(1 - \sum_{c=1}^{C} \alpha_c^2) \tag{6.7}$$

When labels are uniformly distributed across nodes ($\alpha_c = \frac{1}{C}$), the accuracy of Label-based LIA reaches its maximum:

$$Acc \leq 2hd - d + \frac{N}{N-1}(1 - \frac{1}{C}) \tag{6.8}$$

This theorem provides significant insights into the performance of the Label-based LIA. The accuracy of the attack is influenced by several graph properties:

1. *Homophily ratio (h):* A higher homophily ratio indicates that connected nodes are more likely to share the same label, which increases the attack's accuracy.

2. *Graph density (d):* The density of the graph affects the attack's performance, with denser graphs potentially being more vulnerable.

3. *Class diversity:* The term $1 - \sum_{c=1}^{C} \alpha_c^2$ in the equation represents the class diversity. This metric plays a crucial role in the attack's performance, especially for inferring non-existing links.

The class diversity metric reaches its maximum value of $1 - \frac{1}{C}$ when the distribution of nodes across classes is uniform ($\alpha_c = \frac{1}{C}$ for all c). This

indicates high class diversity. Conversely, when all nodes belong to a single class, the metric reaches its minimum value of zero, indicating no diversity.

In balanced datasets where labels are uniformly distributed, a dataset with a higher number of classes is considered more diverse. This is reflected in the diversity metric $1 - \frac{1}{C}$, which increases as the number of classes C increases. This highlights the importance of considering both the distribution of labels and the number of classes when assessing a graph's vulnerability to Label-based LIA.

The upper bound on accuracy is achieved when labels are uniformly distributed across nodes. This scenario represents the most favorable condition for the attack, as it maximizes the class diversity term.

Proof. We begin by expressing the accuracy in terms of the confusion matrix elements:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\frac{N(N-1)}{2}}, \tag{6.9}$$

where TP, TN, FP, and FN represent True Positives, True Negatives, False Positives, and False Negatives, respectively. The denominator $\frac{N(N-1)}{2}$ represents the total number of possible node pairs in the graph.

These terms are defined as follows:

- **True Positives (TP):** The number of correctly predicted edges that exist between nodes with the same label:

$$TP = |\{(i, j) \mid e_{ij} = 1 \wedge y_i = y_j\}| \tag{6.10}$$

- **True Negatives (TN):** The number of correctly predicted non-edges between nodes with different labels:

$$TN = |\{(i, j) \mid e_{ij} = 0 \wedge y_i \neq y_j\}| \tag{6.11}$$

- **False Positives (FP):** The number of incorrectly predicted edges for pairs of nodes with the same label:

$$FP = |\{(i, j) \mid e_{ij} = 0 \wedge y_i = y_j\}| \tag{6.12}$$

- **False Negatives (FN):** The number of incorrectly predicted non-edges between nodes with different labels:

$$FN = |\{(i, j) \mid e_{ij} = 1 \wedge y_i \neq y_j\}| \tag{6.13}$$

Using the homophily ratio definition, we express TP as:

$$TP = h|E| \tag{6.14}$$

For TN, we derive:

$$\begin{aligned} TN &= |\{(i,j) \mid e_{ij} = 0 \wedge y_i \neq y_j\}| \\ &= |\{(i,j) \mid e_{ij} = 0\}| - |\{(i,j) \mid e_{ij} = 0 \wedge y_i = y_j\}| \\ &= \frac{N(N-1)}{2} - |E| - FP \end{aligned} \tag{6.15}$$

Here, $|\{(i,j) \mid e_{ij} = 0\}|$ represents the number of non-edges, which is equal to the total number of possible edges in the graph ($\frac{N(N-1)}{2}$) minus the number of existing edges ($|E|$).

Developing the term FP, we obtain:

$$\begin{aligned} FP &= |\{(i,j) \mid e_{ij} = 0 \wedge y_i = y_j\}| \\ &= |\{(i,j) \mid y_i = y_j\}| - |\{(i,j) \mid e_{ij} = 1 \wedge y_i = y_j\}| \\ &= |\{(i,j) \mid y_i = y_j\}| - TP \\ &= |\{(i,j) \mid y_i = y_j\}| - h|E| \end{aligned} \tag{6.16}$$

The first term counts combinations of node pairs with the same label, expressed as:

$$|\{(i,j) \mid y_i = y_j\}| = \sum_{c=1}^{C} \binom{N_c}{2}, \tag{6.17}$$

where N_c is the number of nodes with label c, and C is the number of labels. Defining $\alpha_c = \frac{N_c}{N}$ as the proportion of nodes with label c, and noting $\sum_{c=1}^{C} \alpha_c = 1$, we derive:

$$\begin{aligned} FP &= \sum_{c=1}^{C} \frac{\alpha_c N(\alpha_c N - 1)}{2} - h|E| \\ &= \frac{N^2}{2} \sum_{c=1}^{C} \alpha_c^2 - \frac{N}{2} - h|E| \end{aligned} \tag{6.18}$$

Substituting the expression for FP into TN, we obtain:

$$
\begin{aligned}
TN &= \frac{N(N-1)}{2} - |E| - FP \\
&= \frac{N(N-1)}{2} - |E| - \left(\frac{N^2}{2} \sum_{c=1}^{C} \alpha_c^2 - \frac{N}{2} - h|E| \right)
\end{aligned} \tag{6.19}
$$

Substituting the expressions for TP and TN into the accuracy formula and applying the graph density definition $d = \frac{2|E|}{N(N-1)}$, we deduce the accuracy of the attack as follows:

$$
\begin{aligned}
Acc &= \frac{TP + TN}{\frac{N(N-1)}{2}} \\
&= \frac{h|E| + \frac{N(N-1)}{2} - |E| - \left(\frac{N^2}{2} \sum_{c=1}^{C} \alpha_c^2 - \frac{N}{2} - h|E| \right)}{\frac{N(N-1)}{2}} \\
&= \frac{4h|E|}{N(N-1)} + 1 - \frac{2|E|}{N(N-1)} - \frac{N}{N-1} \sum_{c=1}^{C} \alpha_c^2 + \frac{1}{N-1} \\
&= 2h \frac{|E|}{\frac{N(N-1)}{2}} - \frac{|E|}{\frac{N(N-1)}{2}} - \frac{N}{N-1} \sum_{c=1}^{C} \alpha_c^2 + 1 + \frac{1}{N-1} \\
&= 2hd - d - \frac{N}{N-1} \sum_{c=1}^{C} \alpha_c^2 + \frac{N}{N-1} \\
&= 2hd - d + \frac{N}{N-1} (1 - \sum_{c=1}^{C} \alpha_c^2)
\end{aligned} \tag{6.20}
$$

To establish the upper bound of accuracy, we apply the lower bound $\frac{1}{C} \leq \sum_{c=1}^{C} \alpha_c^2$, which is a direct application of the $L_1 - L_2$ norm inequality:

$$
\begin{aligned}
C \sum_{c=1}^{C} \alpha_c^2 &\geq \left(\sum_{c=1}^{C} \alpha_c \right)^2 = 1^2 = 1 \\
\sum_{c=1}^{C} \alpha_c^2 &\geq \frac{1}{C}
\end{aligned} \tag{6.21}
$$

Finally, we prove that the upper bound on the accuracy Acc, using the lower bound above, is:

$$Acc \leq 2hd - d + \frac{N}{N-1}(1 - \frac{1}{C}) \tag{6.22}$$

This upper bound is attained when $\alpha_c = \frac{1}{C}$ for all c, corresponding to a uniform distribution of labels across nodes. This concludes our proof, demonstrating the derived upper bound for the accuracy of the attack. □

This theorem and its proof provide a comprehensive understanding of the factors influencing the accuracy of the Label-based LIA. It highlights the interplay between graph structure (through homophily and density) and label distribution (through class diversity) in determining the attack's effectiveness. This analysis can be valuable for assessing the vulnerability of different types of graphs to such attacks and for developing appropriate defense strategies.

6.3.1 Equivalence of Output-based and Label-based LIA

When a model is well-trained, its prediction output highly correlates with the true label. Consequently, the performance of the Output-based LIA is closely related to that of the Label-based LIA. In this section, we establish a direct equivalence between these two attacks under specific conditions.

Theorem 7 (Equivalence of Output-based and Label-based LIA)**.** For a Graph Neural Network (GNN) trained with cross-entropy loss for a binary classification task, if each training sample's loss l_i satisfies $l_i \leq -\log\left(\frac{3-\sqrt{3}}{2}\right)$, then the Output-based LIA with threshold $\tau = \frac{\sqrt{3}}{2}$ infers the same graph as the Label-based LIA.

This theorem provides a precise condition under which the Output-based LIA, which relies on model predictions, produces identical results to the Label-based LIA, which uses true labels. The significance of this result lies in its demonstration that, under certain circumstances, an attacker can infer the same information from well-trained model outputs as they could from the true labels, without requiring direct access to those labels.

Proof. Consider a binary classification task with classes c_1 and c_2. Let θ represent the minimum probability assigned to the true label of a node. We define:

- $P^1_{max} = [1, 0]$: the maximum posterior probability for class c_1
- $P^1_{min} = [\theta, 1 - \theta]$: the minimum θ posterior probability for class c_1

- $P^2_{min} = [1-\theta, \theta]$: the minimum θ posterior probability for class c_2

Our goal is to find θ such that the cosine similarity between P^1_{max} and P^1_{min} (same class) is greater than the cosine similarity between P^1_{min} and P^2_{min} (different classes). This condition ensures that the Output-based LIA correctly distinguishes between same-class and different-class node pairs.

The cosine similarity between two vectors A and B is given by $\text{Cos}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$.

For the pair (P^1_{max}, P^1_{min}), the cosine similarity is:

$$\begin{aligned} Cos(P^1_{max}, P^1_{min}) &= \frac{1 \cdot \theta + 0 \cdot (1-\theta)}{\sqrt{1^2 + 0^2}\sqrt{\theta^2 + (1-\theta)^2}} \\ &= \frac{\theta}{\sqrt{\theta^2 + (1-\theta)^2}} \end{aligned} \tag{6.23}$$

For the pair (P^1_{min}, P^2_{min}), the cosine similarity is:

$$\begin{aligned} Cos(P^1_{min}, P^2_{min}) &= \frac{\theta \cdot (1-\theta) + (1-\theta) \cdot \theta}{\sqrt{\theta^2 + (1-\theta)^2}\sqrt{(1-\theta)^2 + \theta^2}} \\ &= \frac{2\theta(1-\theta)}{\theta^2 + (1-\theta)^2} \end{aligned} \tag{6.24}$$

To find θ such that these similarity conditions are equal (representing the boundary condition for our inequality), we equate the two cosine similarities:

$$\frac{\theta}{\sqrt{\theta^2 + (1-\theta)^2}} = \frac{2\theta(1-\theta)}{\theta^2 + (1-\theta)^2} \tag{6.25}$$

Solving this equation yields $\theta = 0$ and $\theta = \frac{3}{2} - \frac{\sqrt{3}}{2}$. The solution $\theta = 0$ is not practical as it does not represent a valid probability for class prediction. Thus, we consider $\theta = \frac{3}{2} - \frac{\sqrt{3}}{2}$, which signifies the minimum probability that must be assigned to the true class of a node to satisfy our condition.

For a model trained with cross-entropy loss, the loss of a sample is given by $l_i = -\log(p_t)$, where p_t is the probability assigned to the true label of the node. Therefore, our condition on θ translates to a condition on the sample loss:

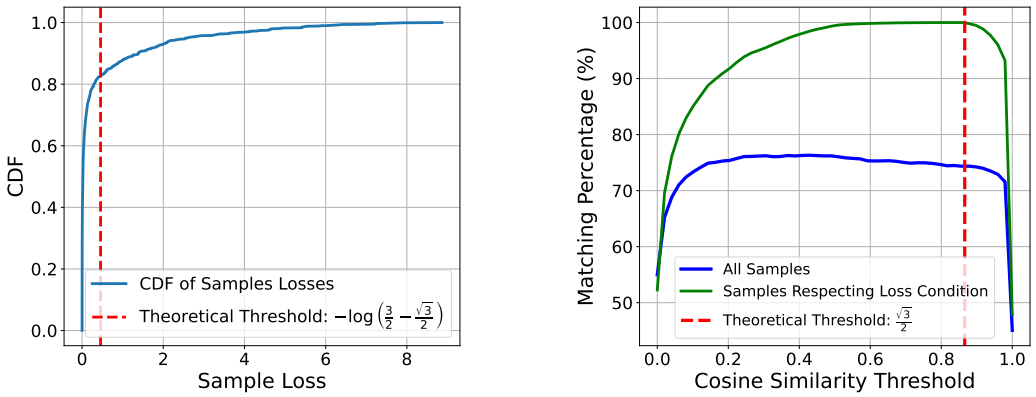
$$l_i \leq -\log\left(\frac{3}{2} - \frac{\sqrt{3}}{2}\right) \tag{6.26}$$

This result establishes a threshold for the probability assigned to the true class, ensuring that inter-class prediction similarities are always greater than

intra-class prediction similarities. To achieve identical inference outcomes between Output-based and Label-based LIA, we incorporate our solution θ into one of the cosine similarity equations. This substitution yields a decision threshold $\tau = \frac{\sqrt{3}}{2}$, ensuring both LIA methods infer equivalent graphs. □

The significance of this theorem lies in its ability to precisely quantify the conditions under which an Output-based LIA can achieve the same performance as a Label-based LIA. This result has important implications for privacy in graph neural networks, as it shows that even without direct access to labels, an attacker can potentially infer the same information from well-trained model outputs.

To validate this theoretical result, we conducted an empirical analysis using the Cora dataset. We focused on a subgraph containing the two most populous classes and trained a two-layer GCN with an input GCN layer of 16 hidden units, followed by a ReLU activation, dropout, and an output GCN layer producing two units. The model was trained using the Adam optimizer with a learning rate of 0.01 and weight decay of 5e-4, minimizing the cross-entropy loss over 200 epochs.



(a) CDF of sample losses

(b) Sensitivity analysis of cosine similarity threshold

Figure 6.2: Analysis of sample losses and cosine similarity threshold

Figure 6.2 presents the results of our empirical validation. In Figure 6.2a, we plot the cumulative distribution function (CDF) of the cross-entropy sample losses, highlighting the theoretical condition $-\log\left(\frac{3-\sqrt{3}}{2}\right)$. Figure 6.2b shows a sensitivity analysis of the cosine similarity threshold, plotting the matching percentage of link predictions between the Output-based and Label-based attacks for various thresholds.

The results demonstrate that when only samples respecting the loss condition are considered, the matching percentage between Output-based and Label-based attacks reaches 100% at the theoretical threshold $\tau = \frac{\sqrt{3}}{2}$. This empirical evidence strongly supports our theoretical findings, confirming that pairs of samples with cross-entropy loss below the theoretical threshold result in identical link predictions for both attack methods.

6.3.2 Performance of Gradient-based LIA

While a participating client in a VFGL setting lacks direct access to the prediction output P and the labels $\mathcal{Y}$, the gradient $G_{\mathcal{A}}$ implicitly encodes this information. Previous research on centralized graph learning has demonstrated that gradients can leak label information [40, 82]. In this section, we analyze how the Gradient-based LIA leverages this information leakage and show its close relationship to the Label-based LIA.

To illustrate this relationship, we present a simplified binary classification example that demonstrates how the cosine similarity of gradients in our Gradient-based LIA is heavily dependent on the labels rather than the prediction output.

Example 1 (Binary Classification with Log Loss)**.** *Consider a binary classification task with log loss, where the DNN owned by Party $\mathcal{P}_{\mathcal{Y}}$ consists of a fully connected layer with weights $\mathbf{A}$ and bias $\mathbf{b}$, followed by a sigmoid function σ. Let $H_{\mathcal{A}}$ and $H_{\mathcal{G}}$ be the intermediate representations of parties $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{G}}$ respectively, with H^i denoting the i-th row of matrix H and $\mathcal{Y}_i$ the label of sample i.*

For every sample i, the prediction output $\hat{\mathcal{Y}}_i$ is given by:

$$\hat{\mathcal{Y}}_i = \sigma(\mathbf{A}[H^i_{\mathcal{G}} \;\; H^i_{\mathcal{A}}] + \mathbf{b}) \tag{6.27}$$

The log loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}_i = -\sum_{i=1}^{N} \mathcal{Y}_i \log \hat{\mathcal{Y}}_i + (1 - \mathcal{Y}_i) \log(1 - \hat{\mathcal{Y}}_i) \tag{6.28}$$

The gradient of sample i with respect to $H^i_{\mathcal{A}}$ is:

$$G^i_{\mathcal{A}} = \frac{\partial L}{\partial H^i_{\mathcal{A}}} = \frac{\partial L_i}{\partial H^i_{\mathcal{A}}} = (\hat{\mathcal{Y}}_i - \mathcal{Y}_i)\mathbf{A}' \tag{6.29}$$

where $\mathbf{A}'$ are the weights in $\mathbf{A}$ that apply to $H^i_{\mathcal{A}}$.

From Equation 6.29, we can observe that since $\forall i, 0 \leq \hat{\mathcal{Y}}_i \leq 1$, regardless of the prediction outputs, if two samples i and j have different labels, the cosine similarity of $G^i_{\mathcal{A}}$ and $G^j_{\mathcal{A}}$ is negative. Conversely, if the samples belong to the same label class, the cosine similarity is positive.

This observation leads to the following theorem:

Theorem 8 (Gradient-based LIA Label Dependence)**.** In a binary classification task with log loss, the Gradient-based LIA will infer no link between two samples if and only if they have different labels, regardless of their prediction outputs.

Proof. Consider two samples i and j. From Equation 6.29, we have:

$$G^i_{\mathcal{A}} = (\hat{\mathcal{Y}}_i - \mathcal{Y}_i)\mathbf{A}' \tag{6.30}$$

$$G^j_{\mathcal{A}} = (\hat{\mathcal{Y}}_j - \mathcal{Y}_j)\mathbf{A}' \tag{6.31}$$

The cosine similarity between these gradients is:

$$\cos(G^i_{\mathcal{A}}, G^j_{\mathcal{A}}) = \frac{(\hat{\mathcal{Y}}_i - \mathcal{Y}_i)(\hat{\mathcal{Y}}_j - \mathcal{Y}_j)}{|(\hat{\mathcal{Y}}_i - \mathcal{Y}_i)||(\hat{\mathcal{Y}}_j - \mathcal{Y}_j)|} \tag{6.32}$$

If $\mathcal{Y}_i \neq \mathcal{Y}_j$, then one of $(\hat{\mathcal{Y}}_i - \mathcal{Y}_i)$ or $(\hat{\mathcal{Y}}_j - \mathcal{Y}_j)$ is always positive and the other is always negative, resulting in a negative cosine similarity. Conversely, if $\mathcal{Y}_i = \mathcal{Y}_j$, both terms have the same sign, resulting in a positive cosine similarity. □

The significance of this theorem lies in its demonstration that the Gradient-based LIA is highly dependent on the true labels of the samples, rather than the model's predictions. This makes the performance of the Gradient-based LIA closely aligned with that of the Label-based LIA, which directly uses the true labels.

This result contrasts with other possible attacks for the participating client, such as Baseline LIA with observations of intermediate representations and features, whose performances are inherently related to the observation dimension. The Gradient-based LIA, being less dependent on the gradient dimension and the features dimension, offers a more robust attack strategy in the VFGL setting.

These analytical findings will be further corroborated by experimental results in Section 6.5, where we demonstrate the strong correlation between the performance of Gradient-based and Label-based LIA across various datasets and model configurations.

6.4 Experimental setup

6.4.1 GNN Model architecture and learning setting

Our study follows the model architecture established in the baseline [104]. The server's model, also called the top model, is designed as a DNN featuring two fully connected layers activated by ReLU functions. To determine the vulnerability of the target/victim party's GNN's architecture to the attack, we utilize GCN, GraphSAGE, and GAT architectures. The GNNs' hop count are set to 2. The GNN implementations are derived from a publicly accessible code[1]. Similarly, the model of the other client, who can be a potential adversary, comprises a DNN with two fully connected layers employing ReLU activation functions. The bottom models, i.e., the two FL clients' models, are set to encode input features into a 16-dimensional latent space as a standard configuration, where the first layer maps first into half the input dimension. Note that, by default, the adversary controls 50% of the features.

We follow the same training protocol in baseline [104], where the VFGL models are trained over 300 epochs, by using a learning rate of 0.001 and by setting the regularization parameters to 0.001. The loss function of choice is cross-entropy, and model parameters are updated using the Adam optimizer. All experiments are conducted under identical settings but with different random seeds, repeated five times to calculate and report the average and standard deviation values of the performance metrics.

6.4.2 Datasets

We utilize seven public datasets for our analysis as Cora [76], Citeseer [76], Amazon Computers (Computer) [119], Amazon Photos (Photo) [119], and Twitch-(FR, DE, and EN) [108] datasets. These are widely recognized as benchmark datasets for evaluating the performance and privacy aspects of GNNs [104, 149, 54, 138].

The Citeseer and Cora datasets are citation networks, where nodes and edges respectively correspond to publications and citations among these publications. Node features consist of the declared keywords in the publications, and class labels represent the research fields of the corresponding publications. The Amazon Computers and Amazon Photos datasets are parts of the Amazon co-purchase graph, where nodes and edges respectively represent products and the actual two products are frequently bought together. Node features are bag-of-words representations of the corresponding product reviews, and class labels categorize the product types. The Twitch

[1]https://pytorch-geometric.readthedocs.io/

datasets are social network datasets that depict the followership connections between users on the Twitch streaming platform. Node features include users' preferred games, location, and streaming habits, while class labels indicate whether a streamer uses explicit language.

Table 6.3: Datasets statistics.

Dataset	Nodes	Edges	Features	Classes	Density(%)
Photo	7650	119081	745	8	0.41
Cora	2708	5278	1433	7	0.14
Computer	13752	245861	767	10	0.26
Citeseer	3327	4552	3703	6	0.08
Twitch-DE	9498	157887	128	2	0.35
Twitch-EN	7126	38887	128	2	0.15
Twitch-FR	6551	115941	128	2	0.54

6.4.3 Evaluation Metrics

AUC. The performance of the attack is evaluated by using the area under the ROC curve (AUC). The AUC provides a comprehensive measure of the attack's performance across various decision thresholds, highlighting its threshold-independent nature. An AUC value of 0.5 means that the attack performance is equivalent to random guessing and hence the adversary has no power, as opposed to the case where the AUC approaches 1, the attack becomes successful in inferring graph links.

Accuracy. Accuracy is used to measure the performance of label-based LIA, which uniquely does not require a decision threshold for link prediction unlike the other attacks. For a comparative analysis of label-based LIA against threshold-dependent attacks, accuracy is assessed at the threshold with the highest F1 score. This approach ensures a balance between precision and recall, providing a detailed insight into the effectiveness of each attack strategy.

6.5 Evaluation

6.5.1 Performance of Link Inference Attacks

We first conduct all the attacks across all the datasets in the scenario where Party $\mathcal{P}_{\mathcal{A}}$ owns 50% of the features, i.e., the size of $X_{\mathcal{A}}$ is equivalent to the size of $X_{\mathcal{G}}$. We also report the maximum accuracy achieved over the

training epochs for time-dependent attacks, including Gradient-based, Intermediate Representation-based, and Output-based LIA. The accuracy of the LIA are shown in Table 6.4. Note that participating clients can only conduct Gradient-based, Intermediate-Representation-based and Features-based LIA as mentioned in Section 6.2.

Label-based LIA

The performance of Label-based LIA is positively correlated with the homophily ratio h and the class diversity $1-\sum_{c=1}^{C}\alpha_c^2$, confirming the analytical results in Section 6.3. Due to the low density of the datasets (Table 6.3), the performance of the attack is primarily influenced by the class diversity metric (the third term in Eq. 6.7). The variability in results observed across the 5 trials stems from the random partitioning of nodes into training and testing sets.

Prediction Output-based LIA

For the datasets Photo, Cora, Computer, and Citeseer, the output-based LIA demonstrates similar performance compared to the Label-based LIA, with differences within 2 percentage points (p.p.). This similarity is due to the well-trained model yielding prediction outputs that strongly align with the true labels. For Twitch datasets, the performance displays a weaker correlation with the Label-based attack, likely due to the model being less pertinent, as observed in [138, 149]. Moreover, the links within Twitch datasets are primarily associated with features rather than labels, leading to the prediction output surpassing the performance of the Label-based attack.

Gradient-based LIA

Comparison with Label-based LIA. Our experimental results confirm that the observation from the simplified example in Section 6.3.2 remains applicable in a more intricate DNN scenario when the dataset has a high homophily ratio (>0.7) and class diversity (>0.7). The accuracy difference between Gradient-based and Label-based attacks is within 1.7 p.p. for these datasets.

To further investigate this observation, we examine the predictions of both attacks, focusing on True Positives (TP) and False Positives (FP). We conduct a comparative analysis on a subgraph of the Amazon-Photo dataset, comprising 10,000 positive pairs (unlinked pairs) and 10,000 negative pairs (linked pairs). The experiment (Figure 6.3) reveals a significant overlap in link predictions between the two attacks, suggesting that the success of

Table 6.4: Accuracy of link inference attacks across datasets using GCN architecture. Bold numbers indicate highest accuracy among client-side attacks (first three columns). Participating clients can only conduct these three attacks.

Datasets	h	Class diversity	Attack Methods				
			Gradients	Inter-Reps	Features	Labels	Prediction output
Photo	0.83	0.84	**82.12 ± 1.32**	67.61 ± 2.76	63.58 ± 0.42	83.76 ± 0.10	81.80 ± 0.68
Cora	0.81	0.82	**81.71 ± 0.21**	65.77 ± 1.19	71.34 ± 1.95	81.74 ± 0.15	80.14 ± 0.58
Computer	0.78	0.79	**78.23 ± 1.26**	66.46 ± 0.40	63.75 ± 0.86	79.35 ± 0.05	78.82 ± 0.70
Citeseer	0.74	0.82	**82.76 ± 0.38**	73.53 ± 2.58	82.65 ± 0.70	82.14 ± 0.02	79.64 ± 0.64
Twitch-DE	0.64	0.48	**58.76 ± 0.22**	55.35 ± 1.47	56.76 ± 0.47	48.02 ± 0.03	58.61 ± 0.01
Twitch-EN	0.60	0.50	53.28 ± 0.25	**54.63 ± 0.79**	54.37 ± 0.80	49.65 ± 0.10	53.51 ± 0.44
Twitch-FR	0.55	0.47	50.89 ± 0.66	50.16 ± 1.09	**56.04 ± 0.45**	46.68 ± 0.20	49.72 ± 0.51

gradient-based LIA is not coincidental but rather indicative of its reliance on label information for link prediction.

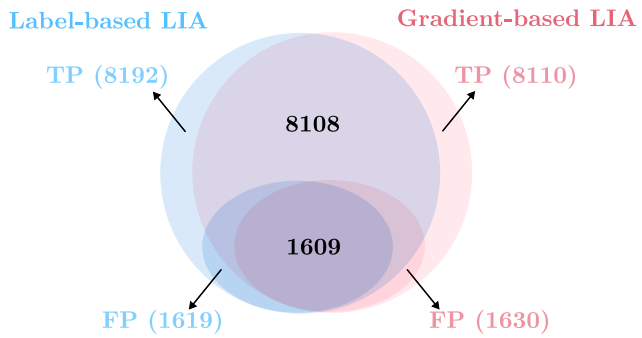


Figure 6.3: Overlap of predictions between gradient-based and label-based LIA in Photo dataset.

Comparison with Prediction Output-based LIA. For datasets with lower homophily ratios and class diversity, our gradient-based LIA performs similarly to the Prediction Output LIA (accuracy differences within 0.3 p.p.). While not significantly improving accuracy, our attack offers greater practicality as it adheres to strict VFL protocol without additional assumptions.

Comparison with Intermediate-representation and feature-based LIA. The Gradient-based LIA outperforms the other two attacks in 5 out of 7 datasets. Particularly for the Photo, Cora, and Computer datasets, the attack performance is enhanced by at least 10 p.p. This improvement can be attributed to the fact that the links in these datasets are more label-related than feature-related. For the Citeseer dataset, the Feature-based LIA performance is close to the Gradient-based LIA performance, given that Citeseer exhibits a unique property where the links are both label-related and feature-related. For the Twitch datasets, where the links are primarily associated with features, the Gradient-based LIA still demonstrates comparable performance.

Overall, the Gradient-based LIA demonstrates comparable results in both label-related and feature-related scenarios, while Intermediate-representation, Feature-based and Label-based attacks fail in one of these scenarios. It also outperforms Prediction output LIA by 0.75 p.p. on average, even though Prediction output LIA requires a stronger adversary. In real-world applications, where the adversary may lack prior knowledge regarding whether the dataset is more label-related or feature-related, the Gradient-based attack stands out as the optimal choice for the adversary.

6.5.2 Ablation Studies

Impact of Training Epochs on Gradient-based LIA Performance

We assess the impact of the training epochs on the performance of our gradient-based LIA and compare it with the Intermediate representation-based LIA baseline. Both attacks can be mounted on the client side by party $P_{\mathcal{A}}$ and are inherently time-dependent. Figure 6.4 illustrates how the learning epochs affect the AUC of these two attack strategies.

Our gradient-based LIA outperforms in the initial epochs, though its efficiency decreases as training progresses. This pattern likely emerges because the gradients in the initial epochs are more informative, gradually becoming less distinguishable as the model nears convergence. As the model approaches convergence, the gradients of all nodes start to concentrate around zero, complicating the adversary's task of differentiating between connected and non-connected node pairs, as depicted in Figure 6.6.

On the other hand, the Intermediate representation-based LIA shows modest performance in the first training epochs, with its performance reaching its maximum after the early training epochs. This pattern is attributed to the fact that, during the early epochs, the intermediate representations do not capture the features of nodes due to the adversary's model lack of optimization. Thus, the Intermediate representation-based LIA reaches peak efficiency in the later stages of training. In contrast, our gradient-based LIA achieves its highest performance in the training's early epochs.

Impact of Feature Ratio Owned by the Adversary

We examine the impact of the proportion of features owned by the adversary $P_{\mathcal{A}}$ on the performance of our gradient-based LIA, intermediate representation LIA, and the feature-based LIA. We analyze the success of the attack across varying ratios of adversary-owned features $X_{\mathcal{A}}$, which aids in comparing the attacks and their dependence on the number of features the adversary possesses.

As illustrated in Figure 6.5, we report the peak AUC over time achieved by the attacks across a range of feature ownership ratios, adjusting from the default condition where the adversary controls 50% of the features, ranging from 10% to 90% of the entire set.

The gradient-based LIA outperforms the intermediate representation-based and feature-based LIA when less than half of the features are owned by the adversary, except for the Twitch-FR dataset. This is mainly due to the low homophily and class diversity, while the features are more representative of the links in the graph. However, the intermediate representation-based

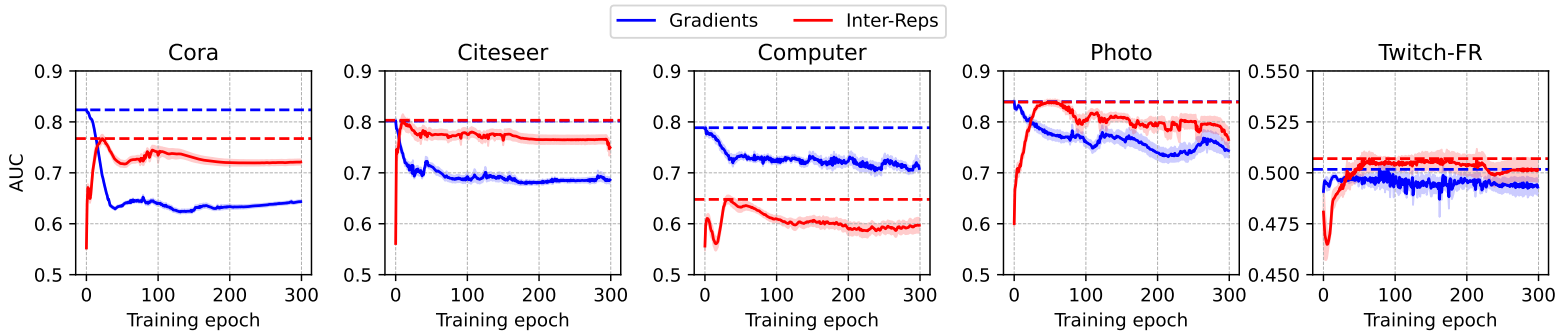


Figure 6.4: Evolution of the AUC for Gradient-based LIA (blue) and Intermediate-representation LIA (red) Over Time. The horizontal dashed lines indicate the maximum AUC achieved by the two attacks. Attacks were conducted at each training epoch, across five runs, with the mean and standard deviation of the AUCs reported.

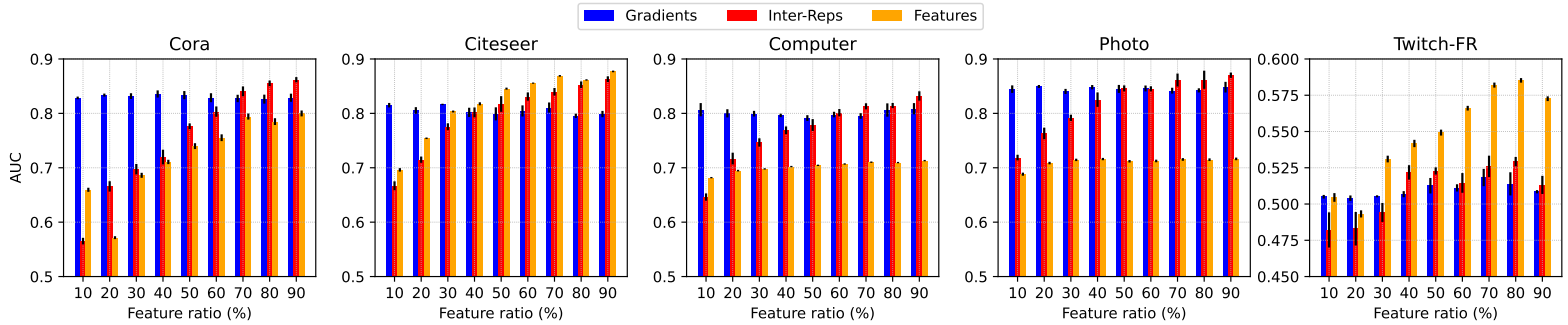


Figure 6.5: Comparison of AUC between Gradient-based LIA (blue), Intermediate Representations-based LIA (red), and Feature-based LIA (orange) across different feature ratios of the adversary. The maximum AUC achieved by both attacks during training is reported, alongside the mean and standard deviation of these AUCs across five runs.

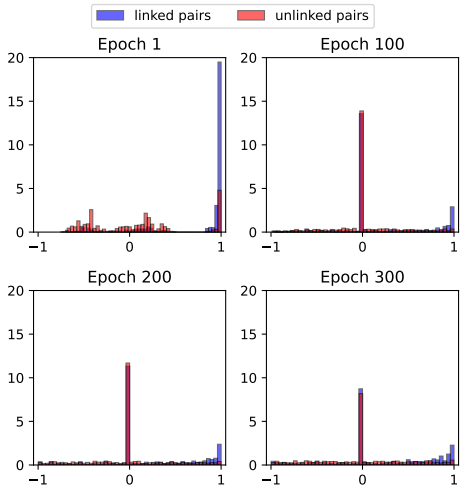


Figure 6.6: Distribution of the cosine similarities of gradients among linked pairs and unlinked pairs during different training epochs.

LIA performs better at higher feature ratios. This phenomenon is attributed to the richness of intermediate representations when a larger number of features is available, leveraging the correlation between features and the connections within the target graph.

The intermediate representation-based LIA's AUC increases proportionally with the number of features the adversary controls. In contrast, the gradient-based LIA's AUC remains relatively unaffected by changes in the number of adversary-owned features. This stability stems from the fact that gradients rely not only on the adversary's intermediate representations, which are impacted by the features owned, but also on the labels of the training nodes as discussed in section 6.3.2.

The same phenomenon of feature size dependence on the performance of feature-based LIA is observed, except for the Photo and Computer datasets. This may be attributed to the sparsity of the features compared to other datasets. In the Photo and Computer datasets, the feature sparsity is relatively higher than in the other datasets as indicated in [104], which means that even 10% of the features is sufficient to reach the maximum performance in inferring the links of these datasets.

Impact of the Number of Parties on the Performance of Gradient-based LIA

We analyze the sensitivity of the number of parties on the performance of our gradient-based attack in varying multi-party settings. We extend our VFGL

protocol by adding more participants, assuming these additional participants are benign and contribute some features. We maintain one adversary and one victim party.

We adopt the same multiparty protocol utilized in the baseline study. Specifically, we fix the adversary's feature ratio at 20%. The remaining features are evenly distributed among the other benign participants, with each having the same feature ratio as the adversary, while the target victim retains the rest of the features. For instance, when there are four parties, the adversary possesses the first 20% of features, the benign participants also have 20% each, and the target victim participant holds the remaining 40%. We vary the number of parties from 2 to 5 to ensure each participant has at least 20% of the features.

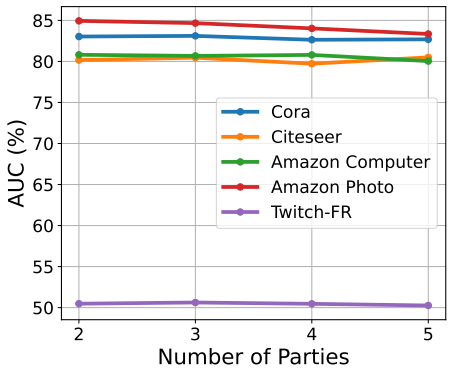


Figure 6.7: Performance analysis with varying number of parties. The results show the AUC values across different datasets as the number of parties changes from 2 to 5.

From Figure 6.7, we observe that the performance of our gradient-based attack is independent of the number of parties. This is primarily because the attack exploits the label information embedded in the gradients, regardless of the number of clients participating in the VFGL setting. It is worth noting that the label-based attack's performance also remains constant across different numbers of clients. This consistency is due to the labels being owned by the server in our VFL setting (Figure 6.1), ensuring that the label information available to the server remains unchanged regardless of the number of participating parties in the VFGL protocol.

Impact of GNN's Architecture on the Performance of Gradient-based LIA

We study the influence of the architecture of the attacked GNN on the accuracy of our gradient-based LIA. For the architectures under examination, we have included GAT and GraphSAGE, as they are some of the most well-known architectures used in the literature.

GATs are distinguishable by their ability to assign varying levels of importance to nodes in a neighborhood, using attention mechanisms. This approach allows GATs to focus on the most relevant parts of the graph structure during the learning process. As observed in our results (see Table 6.5), GATs generally exhibit higher values across various datasets, indicating a greater susceptibility to LIA. This slight increased vulnerability may be attributed to the attention mechanism in GATs that learns the attention coefficients of the edges, which may results in a slight memorization of the edges of the target graph.

In contrast, GraphSAGE utilizes a neighborhood sampling and aggregation approach to generate node embeddings. This method, as reflected in the table, generally shows lower values compared to GAT, suggesting reduced susceptibility to LIA. The fixed-size neighborhood sampling employed by GraphSAGE potentially obscures some links between nodes. By sometimes missing information about node links, GraphSAGE's approach provides a form of obfuscation against such attacks, making it slightly challenging for attackers to accurately infer links.

The performance of GCNs, which falls between GAT and GraphSAGE in our study, suggests a moderate level of susceptibility to LIA. It indicates that while GCNs do learn node connections, they neither reveal as much detail about the links in the graph as GATs nor obscure connections as GraphSAGE.

Table 6.5: Accuracy of gradient-based link inference attack on different GNN architectures

Dataset	GAT	GCN	GraphSAGE
Photo	84.27 $\pm$ 0.62	82.12 $\pm$ 1.32	82.34 $\pm$ 1.31
Cora	82.23 $\pm$ 0.97	81.71 $\pm$ 0.21	81.40 $\pm$ 0.65
Computer	79.38 $\pm$ 0.39	78.23 $\pm$ 1.26	78.06 $\pm$ 1.32
Citeseer	83.40 $\pm$ 1.15	82.76 $\pm$ 0.38	82.35 $\pm$ 0.15

Impact of Model Complexity on the Performance of Gradient-based LIA

We investigate the influence of model complexity on the accuracy of our gradient-based LIA by implementing ResNet-like architectures [53] for both the adversary's and server's models. This study aims to understand how increasing model complexity affects the attack's effectiveness compared to the simple 2-layer neural network baseline.

As observed in our results (see Table 6.6), the attack maintains high accuracy across different levels of model complexity on the Cora dataset. The baseline 2-layer neural network achieves an accuracy of 81.71% ± 0.21%. Interestingly, as we increase the number of residual blocks, we notice only a slight decrease in attack accuracy. With one residual block, the accuracy remains nearly identical at 81.70% ± 0.21%, while with four residual blocks, it decreases marginally to 81.26% ± 0.18%.

This trend suggests that our gradient-based LIA exhibits robustness against increases in model complexity. The attack's resilience can be attributed to two key factors. First, even in more complex models, the gradients still carry sufficient information about the underlying graph structure to enable effective link inference. Second, and crucially, our attack leverages label information, which remains present in the gradients regardless of model complexity. This label information continues to provide valuable insights for link inference, contributing to the attack's consistent performance across different architectural complexities.

Table 6.6: Accuracy of gradient-based LIA on ResNet-like architectures (Cora dataset)

Number of Residual Blocks	Accuracy (%)
Baseline	81.71 ± 0.21
1	81.70 ± 0.21
2	81.54 ± 0.19
3	81.47 ± 0.23
4	81.26 ± 0.18

6.6 Defense Strategies

To mitigate the vulnerabilities exposed by the link inference attacks, we evaluate two types of defense mechanisms: edge perturbation and label perturbation. The edge perturbation aims to obscure the graph structure, while the label perturbation addresses the core issue of label leakage. For edge perturbation, we employ Lapgraph [138], a differential privacy (DP) mechanism that guarantees edge-level DP by adding noise to the graph structure. For label perturbation, we develop a novel approach using quadratic optimization to strategically obfuscate labels.

In our VFGL setting, the edge-level perturbation (Lapgraph) is implemented by the client, which owns the graph structure, while the label perturbation is applied by the server, which possesses the labels. We analyze how these distinct defenses impact each of the previously introduced attacks.

6.6.1 Lapgraph

Lapgraph [138] is a defense mechanism that applies differential privacy at the edge level by perturbing the adjacency matrix of the graph. It adds Laplace noise to the adjacency matrix, controlled by a privacy parameter ε, followed by a binarization process. This effectively alters the graph structure by potentially adding fake edges and removing real ones. In our implementation, we vary ε from 1 to 10 to demonstrate the defense's effectiveness across different privacy levels.

Figure 6.8 shows the results of Lapgraph defense on the Computer, Photo, Cora, and Citeseer datasets, along with a random guessing baseline for test accuracy of the model. The results indicate that the Lapgraph defense does not effectively mitigate the attacks' effectiveness across all datasets, particularly for our introduced gradient-based attack. The gradient-based attack remains highly resilient, showing minimal to no reduction in accuracy across different ε values. This resilience can be attributed to the fact that the gradient information, which is closely tied to the label information, remains largely unaffected by the graph structure perturbation. For instance, on the Cora dataset at $\varepsilon = 6$, the gradient-based attack accuracy is 82.24%, which is even slightly higher than the baseline of 81.71% without any defense.

The prediction output attack follows a similar pattern to the gradient-based attack, with only a marginal decrease in effectiveness across all datasets. On Cora at $\varepsilon = 6$, its accuracy is 80.96%, compared to the baseline of 80.14%. Interestingly, the intermediate representation attack exhibits an unexpected inverse behavior across all datasets, showing increased performance as ε decreases. For Cora at $\varepsilon = 6$, its accuracy rises to 76.89% from the baseline

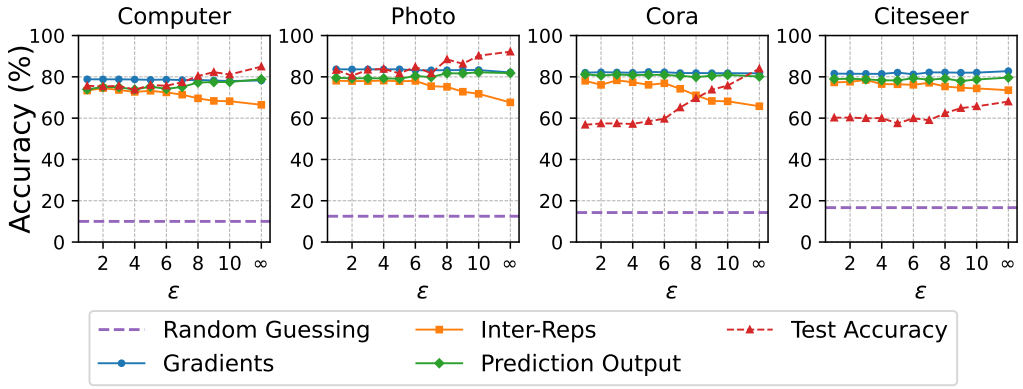


Figure 6.8: Lapgraph defense results. $\varepsilon = \infty$ represents no defense; lower ε values indicate stronger privacy protection.

of 65.77%. We speculate that this counterintuitive result occurs because as the graph edges become noisier, the server model pays more attention to the adversary model than to the victim GNN model. This shift leads to the adversary model becoming a stronger learner about the labels and features of the nodes, hence making its intermediate representation a better signal for the edges of the graph, thus increasing the attack's accuracy. This phenomenon aligns with observations in [104], where perturbation of the victim party's intermediate representations as a defense mechanism inadvertently resulted in strengthening the intermediate representation-based attack of the adversary model.

It is important to note that the label-based attack remains unaffected by this defense. This is because Lapgraph only perturbs the graph structure and does not alter the label information, which is the primary source of information for this attack. The same goes for features-based attack, as Lapgraph does not modify the node features.

While the defense shows minimal impact on the attacks' effectiveness, there is a significant decrease in test accuracy as ε decreases, indicating a substantial utility loss across all datasets. On Cora at $\varepsilon = 6$, the test accuracy drops to 59.66% from the baseline of 83.97%. This underscores the limitation of edge perturbation as a defense mechanism against our proposed attacks, which primarily exploit label information leakage rather than graph structural properties.

6.6.2 Label Perturbation

Label perturbation is a defense mechanism that directly addresses the issue of label leakage by strategically obfuscating a portion of the labels. This approach operates with a budget B, representing the percentage of labels to be changed. We implement this defense by formulating a quadratic optimization problem to find the optimal class proportions that minimize the attack accuracy derived in our theorem 6, subject to the budget constraint.

The label perturbation defense is implemented in two steps: label proportion optimization and label redistribution. Given the accuracy formula for Label-based LIA (Equation 6.7), our experimental observations (Table 6.4) show that terms involving homophily ratio (h) and density (d) are negligible due to the low density of typical graphs in our scenarios. We can thus approximate the accuracy as:

$$Acc \approx \frac{N}{N-1}(1 - \sum_{c=1}^{C} \alpha_c^2) \tag{6.33}$$

To minimize this approximated accuracy and reduce the effectiveness of the Label-based LIA, our objective becomes maximizing $\sum_{c=1}^{C} \alpha_c^2$.

Algorithm 8 optimizes label proportions to maximize $\sum_{c=1}^{C} \alpha_c^2$ within the given budget B using quadratic programming. It takes as input the initial label proportions α_c^{init} and the budget B, and outputs the optimized label proportions α_c^*. The algorithm solves an optimization problem that maximizes $\sum_{c=1}^{C} \alpha_c^2$ subject to constraints ensuring the sum of proportions equals 1, the total change in proportions is within the budget, and all proportions are non-negative.

Algorithm 8 Label Proportion Optimization

Require: Initial label proportions α_c^{init}, budget B
Ensure: Optimized label proportions α_c^*
Solve the following optimization problem:
$\max_{\alpha_c} \sum_{c=1}^{C} \alpha_c^2$
subject to:
  $\sum_{c=1}^{C} \alpha_c = 1$
  $\sum_{c=1}^{C} \max(0, \alpha_c^{init} - \alpha_c) \leq B$
  $\alpha_c \geq 0, \forall c \in \{1, \ldots, C\}$
return α_c^*

Algorithm 9 then redistributes labels to match these optimized proportions through iterative balancing between classes. It takes as input the current labels y and the optimized proportions α^*, and outputs the obfuscated

labels y'. The algorithm first calculates the target number of nodes for each class based on α^*, then determines which classes need more nodes (C^+) and which need fewer nodes (C^-). It then iteratively flips labels from classes in C^- to classes in C^+ until either the target class has enough nodes or the source class has no more nodes to give.

Algorithm 9 Label Redistribution

Require: Current labels y, optimized proportions α^*
Ensure: Obfuscated labels y'

```
Calculate target number of nodes for each class based on α*
Determine classes needing more nodes (C+) and fewer nodes (C−)
for each class c_t ∈ C+ do
    for each class c_s ∈ C− do
        Flip labels from c_s to c_t until:
            - c_t has enough nodes, or
            - c_s has no more nodes to give
    end for
end for
return y'
```

Figure 6.9 shows the results of label perturbation defense on the Computer, Photo, Cora, and Citeseer datasets, along with a random guessing baseline for test accuracy of the model. From the results, we observe that the label perturbation defense demonstrates significant effectiveness in mitigating various attacks across all datasets. As the budget increases from 0.05 to 0.90, we see a substantial decrease in the accuracy of all attacks, albeit at different rates.

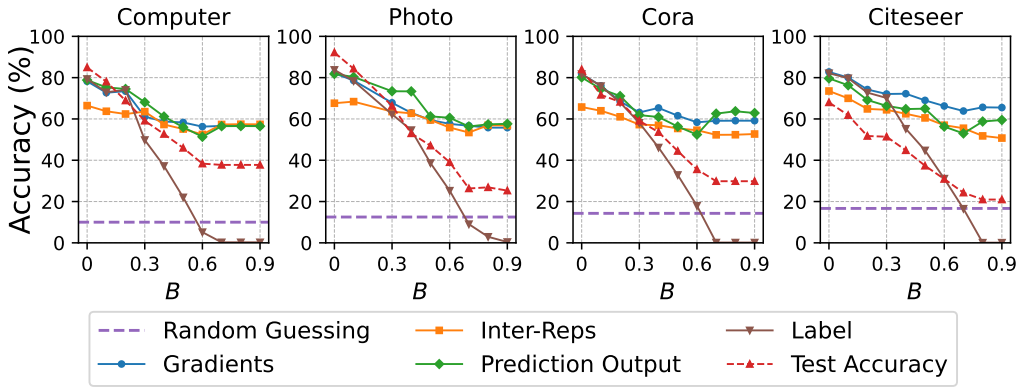


Figure 6.9: Label perturbation defense. $B = 0$ represents no defense; higher B values indicate stronger protection.

The label-based attack is the most affected across all datasets, with its accuracy on Cora dropping from 79.14% at a 0.05 budget to near-zero (0.14%) at a 0.90 budget, compared to the baseline of 81.74% without defense. This dramatic reduction is expected as the defense directly targets the label information that this attack relies on.

The gradient-based attack also shows a significant decrease in accuracy across all datasets. On Cora, it drops from 79.03% at a 0.05 budget to 58.93% at a 0.90 budget, a substantial reduction from the baseline of 81.71%. This substantial reduction reflects the strong correlation between gradient information and labels. The prediction output attack and inter-representation attack show more resilience to the label perturbation defense across all datasets. On Cora, the prediction output attack's accuracy decreases from 77.41% to 64.18%, while the inter-representation attack's accuracy drops from 66.77% to 51.90% as the budget increases from 0.05 to 0.90. These are still significant reductions from their baselines of 80.14% and 65.77% respectively.

Comparing these results to Lapgraph at points where the utility (test accuracy) is similar, we see that label perturbation tends to be more effective across all datasets. For instance, on Cora at a budget of 0.30, label perturbation achieves a test accuracy of 59.12%, which is comparable to Lapgraph's 59.66% at $\varepsilon = 6$. At these points, the gradient-based attack accuracy is 63.14% for label perturbation, compared to 82.24% for Lapgraph. Similarly, the inter-representation attack accuracy is 57.21% for label perturbation, but 76.89% for Lapgraph. The label perturbation defense does impact the system's utility across all datasets, with the test accuracy on Cora decreasing from 73.87% at a 0.05 budget to 29.79% at a 0.90 budget, compared to the baseline of 83.97%. However, this trade-off appears more favorable compared to Lapgraph, especially at lower privacy levels where the utility loss is less severe.

6.6.3 Discussion

Our analysis reveals that Lapgraph is largely ineffective against the introduced attacks and reduces system utility. In contrast, label perturbation demonstrates effectiveness against all attack types across all datasets examined, offering a more favorable privacy-utility trade-off at budget values $<$ 0.3. However, label perturbation faces utility degradation when the budget exceeds 0.3, presenting an area for further research.

The superior performance of label perturbation can be attributed to its direct targeting of the primary source of information leakage in our attacks: the labels. By strategically obfuscating a portion of the labels, it disrupts the core mechanism that gradient-based and label-based attacks rely on. This

approach proves more effective than Lapgraph's method of altering the graph structure, which does not address the fundamental issue of label information leakage.

However, the utility degradation at higher budget values highlights a key challenge in defending against these attacks. As we increase the level of label perturbation to enhance privacy, we inevitably impact the model's ability to learn from the data, resulting in decreased test accuracy. This trade-off between privacy and utility is a central issue in privacy-preserving machine learning and warrants further investigation.

Future work could focus on mitigating utility loss at higher budget values, potentially by combining approaches or developing more refined label perturbation strategies. For instance, one could explore adaptive perturbation methods that adjust the level of obfuscation based on the sensitivity of different labels or nodes. Another avenue could be to investigate techniques that preserve certain critical relationships in the data while perturbing others, aiming to maintain utility while still providing strong privacy guarantees.

Overall, our results indicate that addressing label leakage through targeted perturbation is more effective than altering graph structure in defending against our proposed attacks. This finding underscores the importance of considering the specific mechanisms of information leakage when designing defense strategies for privacy-preserving machine learning systems, particularly in the context of GNNs in VFGL systems.

6.7 Conclusion

This study has introduced novel link inference attacks on VFL systems, specifically focusing on gradient-based and label-based methods. Our research has revealed that gradients in VFL can inadvertently disclose link information, primarily due to the embedded label information within these gradients. To explore this relationship further, we developed a label-based link inference attack.

Our comprehensive evaluation has confirmed that both the gradient-based and label-based attack methods perform similarly, lending support to our initial hypothesis. These findings indicate that these attacks surpass alternative LIA that rely on model predictions and intermediate representations. This superiority is particularly evident in scenarios where the adversary controls a smaller proportion of the features, highlighting the robustness of our proposed attacks.

The analytical framework we developed provides theoretical guarantees for the performance of the label-based LIA, taking into account common

graph statistics such as density, homophily ratio, and class diversity. This framework not only offers insights into the vulnerabilities of target graphs against LIA but also serves as a valuable tool for estimating the potential performance of gradient-based attacks.

Our research also explored the temporal aspects of these attacks, revealing that the gradient-based LIA is most effective in the early stages of training. This finding has important implications for the timing and implementation of defensive measures in VFL systems.

To address the vulnerabilities exposed by our attacks, we proposed and evaluated two defense mechanisms: edge perturbation using Lapgraph and a novel label perturbation approach. Our analysis revealed that Lapgraph, while effective in some scenarios, is largely ineffective against our introduced attacks and significantly reduces system utility. In contrast, our proposed label perturbation defense demonstrated effectiveness against all attack types across all datasets examined, offering a more favorable privacy-utility trade-off at lower privacy budgets.

These findings underscore the importance of addressing label leakage as a fundamental issue in protecting graph privacy within VFL systems. Our label perturbation approach, by directly targeting the source of information leakage, provides a more effective defense strategy compared to methods that focus solely on altering graph structure.

However, our research also highlights the ongoing challenges in this field. While our label perturbation defense shows promise, it faces utility degradation at higher privacy budgets, presenting an area for further research. Future work could focus on mitigating this utility loss, potentially by combining approaches or developing more refined label perturbation strategies.

In conclusion, this study contributes to the growing body of knowledge on privacy in federated learning systems, particularly in the context of graph data. By exposing new vulnerabilities and proposing effective countermeasures, we hope to inspire further research and development in this critical area. As VFL systems continue to gain prominence in various domains, ensuring robust privacy protections while maintaining utility will remain a key challenge for researchers and practitioners alike.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we study the privacy vulnerabilities in machine learning models, particularly focusing on dimensionality reduction techniques, graph-structured data, and federated learning systems. We observed that existing machine learning methods like Principal Component Analysis (PCA) and Graph Neural Networks (GNNs), as well as federated learning frameworks, suffer from significant privacy risks when deployed in real-world applications. Specifically, these methods are exposed to Membership Inference Attack (MIA) and Link Inference Attack (LIA), which can compromise the privacy of individual data points and relational information. Therefore, this thesis proposed solutions that address the limitations of existing work by developing novel privacy attacks to understand the vulnerabilities and by evaluating differential privacy mechanisms as potential defenses.

For dimensionality reduction techniques, we implemented and evaluated the first membership inference attack against PCA. Our attack demonstrated that PCA-transformed data could leak significant private information, especially when the number of samples is small. This revealed a critical vulnerability in PCA, as adversaries could infer whether specific data points were part of the original training dataset. To address this limitation, we explored differentially private PCA algorithms as a defense mechanism. Our evaluation showed that while differential privacy can mitigate the privacy risks in PCA, it often does so at the expense of reduced utility. We analyzed the trade-off between privacy protection and utility, providing insights into the practical application of differentially private PCA in sensitive contexts.

We further developed the Node Injection Link Stealing (NILS) attack as a powerful new link inference attack against GNNs. NILS significantly en-

hances the effectiveness of link inference over existing methods, highlighting the vulnerabilities of GNNs in preserving edge privacy. Our attack improved the F_1 score by 23.75% on the Twitch dataset and more than tripled the performance on the Flickr dataset compared to state-of-the-art attacks. To mitigate this vulnerability, we designed defense strategies based on differential privacy, introducing a LapGraph-based defense specifically tailored to protect against node injection attacks. Our defense mechanism reduced the effectiveness of the NILS attack by approximately 71.9% while incurring only a minimal average utility loss of about 3.2%. This demonstrates that carefully designed differential privacy mechanisms can effectively enhance the privacy of graph-structured data without significantly compromising model performance.

Last but not least, we investigated privacy vulnerabilities in vertical federated graph learning (VFGL) systems. We introduced novel link inference attacks, including gradient-based and label-based methods, showing that gradients in VFGL can inadvertently disclose link information due to embedded label information. Our attacks surpass alternative methods relying on model predictions and intermediate representations, especially when the adversary controls a smaller proportion of the features. To address these challenges, we propose two defense mechanisms: edge perturbation using LapGraph, which is a differential privacy solution, and a novel label perturbation approach that relies on a heuristic method. Our analysis revealed that while LapGraph had limited effectiveness against our attacks and significantly reduced utility, the label perturbation defense was effective against all attack types across all datasets. It offered a favorable privacy-utility trade-off at lower privacy budgets, highlighting the importance of addressing label leakage in protecting graph privacy within federated learning systems.

In summary, this thesis makes the following key contributions:

- **Novel Privacy Vulnerabilities in Fundamental Techniques:** By demonstrating that PCA and GNNs are susceptible to MIA and LIA, we highlighted critical privacy risks in widely used machine learning methods, emphasizing the need for robust privacy-preserving strategies.

- **Novel Privacy Attacks:** We introduced the first MIA against PCA, the NILS attack against GNNs, and new link inference attacks in VFGL systems. These attacks provided deeper insights into how privacy can be compromised in various machine learning contexts.

- **Study of Differentially Private Mechanisms as a Defense Strategies:** We assessed the effectiveness of differential privacy in mitigating the identified privacy risks. Our evaluations provided valuable insights

into the trade-offs between privacy protection and utility, guiding the practical implementation of privacy-preserving techniques.

Through these contributions, we gain a deeper understanding of privacy risks in machine learning and provide practical solutions to enhance data privacy in sensitive applications. Our findings highlight that privacy concerns are pervasive, and as new data structures and learning methods emerge, novel privacy attacks and defense mechanisms require continuous investigation and definition. While differential privacy proves effective in many scenarios, it remains insufficient for addressing all vulnerabilities, especially those stemming from complex and evolving data structures. Therefore, exploring complementary techniques becomes essential for ensuring robust privacy protection. This work underscores the importance of ongoing research in privacy-preserving machine learning to support the safe and secure deployment of models handling sensitive information.

7.2 Future Work

While this thesis has explored privacy vulnerabilities in machine learning models and proposed mitigation strategies, there are several avenues for future research that could build upon our findings. Below, we elaborate on the most concrete and promising directions.

1. **Investigating Potential Correlations Between Vulnerable Samples in PCA and Downstream Tasks**

   In Chapter 4, we introduced a membership inference attack against PCA and analyzed its effectiveness. A potential future direction is to investigate whether there is a correlation between samples that are vulnerable to MIAs in PCA and those that are vulnerable in downstream tasks, such as neural network classifiers trained on PCA-transformed data. Understanding this relationship could lead to more comprehensive privacy-preserving strategies that protect data throughout the machine learning pipeline. This research could involve conducting systematic studies to identify patterns or features that make certain samples more susceptible to privacy attacks and developing methods to mitigate these risks effectively.

2. **Enhancing Differential Privacy Mechanisms for PCA**

   While our evaluation of differentially private PCA algorithms in Chapter 4 shows that they can mitigate MIAs, the trade-off between privacy

and utility remains a challenge. Future research could focus on developing improved differential privacy mechanisms for PCA that offer a better balance between privacy protection and data utility. This might involve exploring alternative noise addition techniques, optimizing the allocation of the privacy budget, or employing adaptive mechanisms that adjust the level of noise based on data sensitivity. Additionally, integrating models that are inherently robust to privacy attacks could complement differential privacy methods to enhance overall protection.

3. **Label Perturbation Defense with Fairness and Adaptive Strategies in Federated Learning**

   In Chapter 6, we proposed a label perturbation defense mechanism to mitigate link inference attacks in vertical federated graph learning systems. To improve this defense, future work could focus on incorporating fairness objectives to ensure that perturbation does not disproportionately affect certain classes, thus maintaining equitable model performance across all groups. Additionally, implementing adaptive perturbation strategies that adjust the level of label noise during training could enhance utility. For example, applying stronger perturbation during the initial epochs—when attacks are most effective—and reducing it in later epochs could help maintain privacy while improving model convergence and overall performance.

4. **Investigating the Impact of Advanced Feature Extractors on Privacy in Graph Neural Networks**

   In our study of link inference attacks against GNNs in Chapter 5, we considered scenarios with conventional node features such as word frequency [76]. An important direction for future research is to explore how different feature extraction methods, particularly advanced language models (LMs) like BERT [23] or GPT [105], impact the privacy of graph structures. In datasets where nodes represent textual information, LMs are often used to generate rich node embeddings [70]. Examining whether these embeddings influence the susceptibility of GNNs to link inference attacks could provide valuable insights into the relationship between node features and privacy risks. Understanding how the use of advanced feature extractors affects attack success rates would enable practitioners to make informed decisions when designing models, potentially selecting or developing feature extraction techniques that enhance privacy protection without compromising performance.

By pursuing these future research directions, we can deepen our understanding of privacy risks in machine learning and develop more effective

strategies to protect sensitive data. Addressing these challenges will build upon the foundation laid by this thesis and contribute to the advancement of privacy-preserving machine learning.

Bibliography

[1] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation). OJ L 119, 4.5.2016, p. 1–88, 2016. Accessed: date.

[2] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in microrna-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 319–330, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3113–3121, Cambridge, MA, USA, 2014. MIT Press.

[4] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[5] C. Blake. Uci repository of machine learning databases. *http://www.ics.uci.edu/~mlearn/MLRepository.html*, 1998.

[6] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, 2005.

[7] R. Brand, J. Domingo-Ferrer, and J.M. Mateo-Sanz. Reference data sets to test and compare sdc methods for protection of numerical microdata. Technical report.

[8] Solenn Brunet, Sébastien Canard, Sébastien Gambs, and Baptiste Olivier. Novel differentially private mechanisms for graphs. *IACR Cryptol. ePrint Arch.*, page 745, 2016.

[9] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. *The Guardian*, Mar 2018.

[10] Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12(null):1069–1109, July 2011.

[12] Kamalika Chaudhuri, Anand D Sarwate, and Kaushik Sinha. A near-optimal algorithm for differentially-private principal components. *Journal of Machine Learning Research*, 14, 2013.

[13] Rama Chellappa, Charles L Wilson, and Saad Sirohey. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 28(4):399–458, 1996.

[14] Chaochao Chen, Jun Zhou, Longfei Zheng, Huiwen Wu, Lingjuan Lyu, Jia Wu, Bingzhe Wu, Ziqi Liu, Li Wang, and Xiaolin Zheng. Vertically federated graph neural network for privacy-preserving node classification. *arXiv preprint arXiv:2005.11903*, 2020.

[15] Rui Chen, Benjamin C. M. Fung, Philip S. Yu, and Bipin C. Desai. Correlated network data publication via differential privacy. *VLDB J.*, 23(4):653–676, 2014.

[16] Tianqi Chen, Markus Pelger, and Jason Zhu. Machine learning in asset pricing. *arXiv preprint arXiv:1904.00745*, 2019.

[17] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR, 2021.

[18] Mauro Conti, Jiaxin Li, Stjepan Picek, and Jing Xu. Label-only membership inference attack against node-level graph neural networks. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, pages 1–12, 2022.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Chem. Biol. Drug Des.*, 297:273–297, 01 2009.

[20] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[21] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. Node-level differentially private graph neural networks. *CoRR*, abs/2111.15521, 2021.

[22] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 123–138. ACM, 2016.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[24] Sayanton V Dibbo. Sok: Model inversion attack landscape: Taxonomy, challenges, and future roadmap. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 439–456. IEEE, 2023.

[25] Ruyi Ding, Shijin Duan, Xiaolin Xu, and Yunsi Fei. Vertexserum: Poisoning graph neural networks for link inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4532–4541, October 2023.

[26] Ilias Driouich, Chuan Xu, Giovanni Neglia, Frederic Giroire, and Eoin Thomas. A novel model-based attribute inference attack in federated learning. In *FL-NeurIPS'22-Federated Learning: Recent Advances and New Challenges workshop in Conjunction with NeurIPS 2022*, 2022.

[27] Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.

[28] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2020.

[29] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '20, page 76–85, New York, NY, USA, 2021. Association for Computing Machinery.

[30] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[31] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[32] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[33] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. 9(3–4):211–407, aug 2014.

[34] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: Optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 11–20, New York, NY, USA, 2014. Association for Computing Machinery.

[35] Úlfar Erlingsson, Ilya Mironov, Ananth Raghunathan, and Shuang Song. That which we call private. *arXiv preprint arXiv:1908.03566*, 2019.

[36] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[37] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[38] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[39] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing. In *23rd USENIX security symposium (USENIX Security 14)*, pages 17–32, 2014.

[40] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Junfeng Zhou, Alex X. Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *USENIX Security Symposium*, 2022.

[41] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[42] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16937–16947. Curran Associates, Inc., 2020.

[43] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[44] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210, New York, New York, USA, 20–22 Jun 2016. PMLR.

[45] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, KDD '16, pages 855–864, New York, NY, USA, 2016. ACM.

[46] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *arXiv preprint arXiv:1809.04788*, 2018.

[47] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[48] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, 2017.

[49] Moritz Hardt and Aaron Roth. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 331–340, 2013.

[50] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.

[51] Michael Hay, Chao Li, Gerome Miklau, and David D. Jensen. Accurate estimation of the degree distribution of private networks. In Wei Wang, Hillol Kargupta, Sanjay Ranka, Philip S. Yu, and Xindong Wu, editors, *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 169–178. IEEE Computer Society, 2009.

[52] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[54] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *USENIX Security Symposium*, pages 2669–2686, 2021.

[55] James B Heaton, Nicholas G Polson, and Jan Hendrik Witte. Deep learning in finance. *arXiv preprint arXiv:1602.06561*, 2017.

[56] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genetics*, 4, 2008.

[57] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[58] 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)*, 54(11s):1–37, 2022.

[59] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1857–1867, New York, NY, USA, 2020. Association for Computing Machinery.

[60] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[61] Thomas Humphries, Simon Oya, Lindsey Tulloch, Matthew Rafuse, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Investigating membership inference attacks under data dependencies. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 473–488. IEEE, 2023.

[62] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. S. Nordholt, K. Spicer, and P.-P. de Wolf. *Statistical Disclosure Control*. 2012.

[63] Hafiz Imtiaz and Anand D. Sarwate. Differentially private distributed principal component analysis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2206–2210, 2018.

[64] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[65] Bargav Jayaraman and David Evans. Are attribute inference attacks just imputation? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1569–1582, New York, NY, USA, 2022. Association for Computing Machinery.

[66] Bargav Jayaraman, Lingxiao Wang, David E. Evans, and Quanquan Gu. Revisiting membership inference under realistic assumptions. *Proceedings on Privacy Enhancing Technologies*, 2021:348 – 368, 2021.

[67] Zhanglong Ji, Zachary C Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

[68] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 259–274, 2019.

[69] Wuxuan Jiang, Cong Xie, and Zhihua Zhang. Wishart mechanism for differentially private principal components analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[70] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[71] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

[72] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[73] Vishesh Karwa and Aleksandra B. Slavkovic. Differentially private graphical degree sequences and synthetic graphs. In Josep Domingo-Ferrer and Ilenia Tinnirello, editors, *Privacy in Statistical Databases - UNESCO Chair in Data Privacy, International Conference, PSD 2012,*

Palermo, Italy, September 26-28, 2012. Proceedings, volume 7556 of *Lecture Notes in Computer Science*, pages 273–285. Springer, 2012.

[74] Yigitcan Kaya and Tudor Dumitras. When does data augmentation help with membership inference attacks? In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5345–5355. PMLR, 18–24 Jul 2021.

[75] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016.

[76] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[77] Gueorgi Kossinets and Duncan J Watts. Empirical analysis of an evolving social network. *science*, 311(5757):88–90, 2006.

[78] Saloni Kwatra, Anna Monreale, and Francesca Naretto. Balancing act: navigating the privacy-utility spectrum in principal component analysis. In *21st International Conference on Security and Cryptography, SECRYPT 2024, Dijon, 8 July 2024 to 10 July 2024*, pages 850–857. Science and Technology Publications, Lda, 2024.

[79] Saloni Kwatra and Vicenç Torra. Data reconstruction attack against principal component analysis. In *International Symposium on Security and Privacy in Social Networks and Big Data*, pages 79–92. Springer, 2023.

[80] Massimo La Rosa, Antonino Fiannaca, Laura La Paglia, and Alfonso Urso. A graph neural network approach for the analysis of sirna-target biological networks. *International Journal of Molecular Sciences*, 23(22), 2022.

[81] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[82] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *ArXiv*, abs/2102.08504, 2021.

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

[84] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3615–3634, 2024.

[85] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *ArXiv*, abs/1802.04889, 2018.

[86] Peihua Mai and Yan Pang. Vertical federated graph neural network for recommender system. In *International Conference on Machine Learning*, pages 23516–23535. PMLR, 2023.

[87] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[88] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[89] Shagufta Mehnaz, Sayanton V Dibbo, Roberta De Viti, Ehsanul Kabir, Björn B Brandenburg, Stefan Mangard, Ninghui Li, Elisa Bertino, Michael Backes, Emiliano De Cristofaro, et al. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4579–4596, 2022.

[90] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2018.

[91] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 05 2017.

[92] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, 2017.

[93] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 634–646, 2018.

[94] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[95] Bao-Ngoc Nguyen, Keshigeyan Chandrasegaran, Milad Abdollahzadeh, and Ngai-Man Man Cheung. Label-only model inversion attacks via knowledge transfer. *Advances in Neural Information Processing Systems*, 36, 2024.

[96] Hiep H. Nguyen, Abdessamad Imine, and Michaël Rusinowitch. Differentially private publication of social graphs at linear cost. In Jian Pei, Fabrizio Silvestri, and Jie Tang, editors, *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015, Paris, France, August 25 - 28, 2015*, pages 596–599. ACM, 2015.

[97] Xiang Ni, Xiaolong Xu, Lingjuan Lyu, Changhua Meng, and Weiqiang Wang. A vertical federated learning framework for graph convolutional network. *arXiv preprint arXiv:2106.11593*, 2021.

[98] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 11–20. IEEE, 2021.

[99] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414, 2018.

[100] Javier Parra-Arnau, Josep Domingo-Ferrer, and Jordi Soria-Comas. Differentially private data publishing via cross-moment microaggregation. *Information Fusion*, 53:269–288, 2020.

[101] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[102] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[103] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The feret evaluation methodology for face-recognition algorithms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, volume 1, pages 137–143. IEEE, 2000.

[104] Pengyu Qiu, Xuhong Zhang, Shouling Ji, Tianyu Du, Yuwen Pu, Jun Zhou, and Ting Wang. Your labels are selling you out: Relation leaks in vertical federated learning. *IEEE Transactions on Dependable and Secure Computing*, pages 1–16, 2022.

[105] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[106] Sofya Raskhodnikova and Adam Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *arXiv preprint arXiv:1504.07912*, 2015.

[107] Benedek Rozemberczki and Rik Sarkar. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *arXiv preprint arXiv:2101.03091*, 2021.

[108] Benedek Rozemberczki and Rik Sarkar. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *CoRR*, abs/2101.03091, 2021.

[109] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *ICML*, 2019.

[110] A. Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *ArXiv*, abs/1806.01246, 2018.

[111] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *CoRR*, abs/1806.01246, 2018.

[112] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994.

[113] Giovanni Sartor, Francesca Lagioia, et al. The impact of the general data protection regulation (gdpr) on artificial intelligence. 2020.

[114] Anand D. Sarwate and Kamalika Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE Signal Processing Magazine*, 30(5):86–94, 2013.

[115] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, Jan 2009.

[116] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

[117] Fanhua Shang, Zhihui Zhang, Tao Xu, Yuanyuan Liu, and Hongying Liu. Principal component analysis in the stochastic differential privacy model. In *Uncertainty in Artificial Intelligence*, pages 1110–1119. PMLR, 2021.

[118] Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*, 2019.

[119] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *ArXiv*, abs/1811.05868, 2018.

[120] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *CoRR*, abs/1610.05820, 2016.

[121] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *JOSA A*, 4(3):519–524, 1987.

[122] Stephen M. Smith, Aapo Hyvärinen, Gaël Varoquaux, Karla L. Miller, and Christian F. Beckmann. Group-pca for very large fmri datasets. *NeuroImage*, 101:738–749, 2014.

[123] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2615–2632, 2021.

[124] Liwei Song, R. Shokri, and Prateek Mittal. Membership inference attacks against adversarially robust deep learning models. *2019 IEEE Security and Privacy Workshops (SPW)*, pages 50–56, 2019.

[125] Elham Tabassi, Kevin J Burns, Michael Hadjimichael, Andres D Molina-Markham, and Julian T Sexton. A taxonomy and terminology of adversarial machine learning. *NIST IR*, 2019:1–29, 2019.

[126] Florian Tramèr, R. Shokri, Ayrton San Joaquin, Hoang M. Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. *ArXiv*, abs/2204.00032, 2022.

[127] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14:2073–2089, 2021.

[128] Matthew A Turk and Alex P Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[129] Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083, 2018.

[130] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[131] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[132] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017.

[133] Kefan Wang, Jing An, Mengchu Zhou, Zhe Shi, Xudong Shi, and Qi Kang. Minority-weighted graph neural network for imbalanced node classification in social networks of internet of people. *IEEE Internet of Things Journal*, 10(1):330–340, 2022.

[134] Xiuling Wang and Wendy Hui Wang. Group property inference attacks against graph neural networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2871–2884, 2022.

[135] Zhaohui Wang, Qi Cao, Huawei Shen, Bingbing Xu, Muhan Zhang, and Xueqi Cheng. Towards efficient and expressive gnns for graph classification via subgraph-aware weisfeiler-lehman. In Bastian Rieck and Razvan Pascanu, editors, *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, page 17. PMLR, 2022.

[136] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Adapting membership inference attacks to GNN for graph classification: Approaches and implications. In James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu, editors, *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, pages 1421–1426. IEEE, 2021.

[137] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Adapting membership inference attacks to gnn for graph classification: approaches and implications. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1421–1426. IEEE, 2021.

[138] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2005–2024. IEEE, 2022.

[139] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page

1307–1322, New York, NY, USA, 2017. Association for Computing Machinery.

[140] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[141] Yi Xie, Yun Xiong, and Yangyong Zhu. Sast-gnn: a self-attention based spatio-temporal graph neural network for traffic prediction. In *Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24–27, 2020, Proceedings, Part I 25*, pages 707–714. Springer, 2020.

[142] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[143] Runhua Xu, Nathalie Baracaldo, and James Joshi. Privacy-preserving machine learning: Methods, challenges and directions. *arXiv preprint arXiv:2108.04417*, 2021.

[144] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[145] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.

[146] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, 2018.

[147] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

[148] Oualid Zari, Javier Parra-Arnau, Ayşe Ünsal, Thorsten Strufe, and Melek Önen. Membership inference attack against principal component analysis. In *International Conference on Privacy in Statistical Databases*, pages 269–282. Springer, 2022.

[149] Oualid Zari, Javier Parra-Arnau, Ayşe Ünsal, and Melek Önen. Node injection link stealing attack. *ArXiv*, abs/2307.13548, 2023.

[150] Oualid Zari, Chuan Xu, and Giovanni Neglia. Efficient passive membership inference attack in federated learning. *arXiv preprint arXiv:2111.00430*, 2021.

[151] Oualid Zari, Chuan Xu, Javier Parra-Arnau, Ayse Unsal, and Melek Onen. Link inference attacks in vertical federated graph learning. In *2024 Annual Computer Security Applications Conference (ACSAC)*, Hawaii, USA, 2024. IEEE. To appear.

[152] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[153] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. Federated graph learning–a position paper. *arXiv preprint arXiv:2105.11099*, 2021.

[154] Jun Zhang, Xiao Zheng, Ying Bi, Li Wang, Hao Zhang, Yiqun Chen, Zhao Li, and Yang Zhang. Differentially private graph neural network for link prediction. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1823–1837, 2021.

[155] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[156] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4543–4560, 2022.

[157] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.

[158] Chengyu Zhou, Yuqi Su, Tangbin Xia, and Xiaolei Fang. Federated multilinear principal component analysis with applications in prognostics. *arXiv preprint arXiv:2312.06050*, 2023.

[159] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[160] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against gans. *arXiv preprint arXiv:2111.07608*, 2021.

[161] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[162] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.