

EC-LDA : Label Distribution Inference Attack against Federated Graph Learning with Embedding Compression

Tong Cheng¹, Jie Fu², Xinpeng Ling¹, Huifa Li¹, Zhili Chen¹

¹East China Normal University ²Stevens Institute of Technology

{tcheng, xpling, huifali}@stu.ecnu.edu.cn, jfu13@stevens.edu, zhlichen@sei.ecnu.edu.cn

Abstract

Graph Neural Networks (GNNs) have been widely used for graph analysis. Federated Graph Learning (FGL) is an emerging learning framework to collaboratively train graph data from various clients. However, since clients are required to upload model parameters to the server in each round, this provides the server with an opportunity to infer each client’s data privacy. In this paper, we focus on *label distribution attacks* (LDAs) that aim to infer the label distributions of the clients’ local data. We take the first step to attack client’s label distributions in FGL. Firstly, we observe that the effectiveness of LDA is closely related to the variance of node embeddings in GNNs. Next, we analyze the relation between them and we propose a new attack named EC-LDA, which significantly improves the attack effectiveness by compressing node embeddings. Thirdly, extensive experiments on node classification and link prediction tasks across six widely used graph datasets show that EC-LDA outperforms the SOTA LDAs. For example, EC-LDA attains optimal values under both Cos-sim and JS-div evaluation metrics in the CoraFull and LastFM datasets. Finally, we explore the robustness of EC-LDA under differential privacy protection.

1 Introduction

1.1 Background

Graph Neural Networks (GNNs), designed to process graph-structured data, have gained significant attention for their effectiveness across various applications including recommendation systems [He *et al.*, 2020], social networks [Fan *et al.*, 2019], and protein interaction prediction [Jha *et al.*, 2022]. GNNs can capture information between neighboring nodes, enhancing the expressiveness of node embeddings and making them highly suitable for real-world applications.

The performance of GNNs requires a large amount of data. However, due to privacy concerns and regulatory restrictions, machine learning platforms cannot access raw data directly, which makes centralized GNN learning challenging. In recent years, a lot of works [He *et al.*, 2021;

Zhang *et al.*, 2021] have integrated Federated Learning (FL) [McMahan *et al.*, 2017] with GNNs, proposing the Federated Graph Learning (FGL). FL is a distributed, privacy-preserving machine learning paradigm that enables clients to train models collaboratively while keeping their local data isolated. It addresses the challenge of data silo, where data is distributed across different sources and cannot be easily combined for joint analysis, by enabling model training without the need to share the raw data.

However, recent studies have shown that local data in FL remains vulnerable to privacy attacks [Zhu *et al.*, 2019; Geiping *et al.*, 2020; Zhao *et al.*, 2020]. One such attack is label distribution inference attacks (LDAs), which aim to infer the label distribution of a client’s local training data by analyzing the gradients shared between clients and the server. This represents a significant privacy threat in FL. For example, if multiple online shopping companies collaborate to train a recommendation system model (as shown in Figure 1), a malicious server with access to the label distribution of a private social network could target specific users, increasing the success rate of fraudulent activities and posing a serious threat to user security.

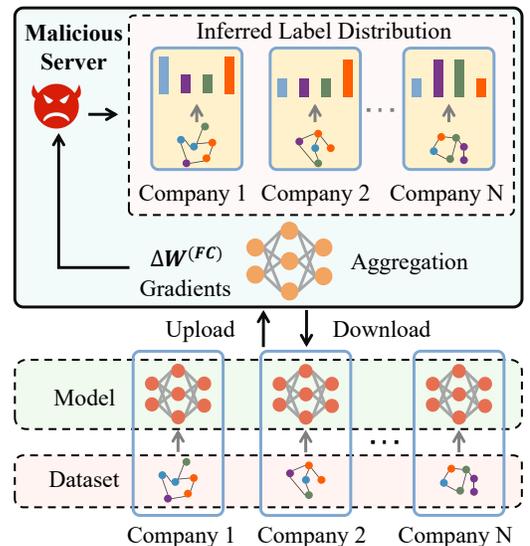


Figure 1: An overview of EC-LDA.

1.2 Previous Works and Problems

Despite existing works on LDAs, there are significant challenges when applying them to FGL. Firstly, these attacks do have certain limitations. For example, [Gu and Bai, 2023; Wainakh *et al.*, 2021] requires the use of real auxiliary datasets to execute the attacks. [Yin *et al.*, 2021] requires the use of non-negative activation functions. Additionally, the performance of attacks in [Ma *et al.*, 2023; Wainakh *et al.*, 2021] declines as the local epochs in FL increase, achieving effective results only with a single local training epoch. These assumptions restrict their applicability in GNN scenarios. Furthermore, the message-passing characteristic among adjacent nodes in GNNs complicates node embeddings, potentially introducing irrelevant details into the model output, thereby making LDA in GNNs more challenging.

1.3 Our Contributions

In this paper, we propose Embedding Compression-Label Distribution Inference Attack (EC-LDA), a novel LDA in FGL. Through exploring the factors behind the poor performance of LDAs in GNNs, we find there are strong correlation between the effectiveness of LDA and the variance of node embeddings. In GNNs, the variance in node embeddings primarily originates from the GNN layers and increases with the number of layers, which in turn degrades the attack performance of LDA. EC-LDA addresses this issue by compressing node embeddings, thereby reducing their variance and enhancing attack performance. Moreover, EC-LDA overcomes the drawbacks of existing methods. It keeps stable performance as training samples and local epochs increase and doesn't need specific activation functions or extra datasets for attacks. Our main contributions are as follows:

- We analyze the relationship between the performance of LDA and the message-passing characteristics of GNNs, and introduce EC-LDA, the first approach that implement efficient LDA on FGL.
- We apply EC-LDA to six graph datasets and conduct extensive experiments targeting both node classification and link prediction tasks. These experiments demonstrate that EC-LDA consistently achieves significant attack performance across various scenarios. EC-LDA achieves a Cos-sim as high as 1.000 under almost all cases.
- We utilize DP-GNN (node-level DP) and label-DP (label-level DP) for local GNN training and evaluate the robustness of EC-LDA under these privacy protection mechanisms.

2 Preliminaries

2.1 Graph Neural Networks

In general, GNNs are designed to process data $G(V, \mathbb{E})$ structured as a graph. Here, V represents the set of nodes and \mathbb{E} represents the adjacency matrix of G . Each node $v_i \in V$ has a feature vector u_i . GNNs generate valuable node embeddings via message-passing, making them suitable for various downstream tasks like node classification and link prediction. In this paper, we consider Graph Convolutional Networks

(GCN) [Kipf and Welling, 2016], Graph Attention Networks (GAT) [Veličković *et al.*, 2017], and GraphSAGE [Hamilton *et al.*, 2017] as target GNNs.

GNNs typically follow the message-passing strategy that updates the features of nodes iteratively by aggregating the features of their neighbors. Typically, a GNN model's h -th layer can be formulated as:

$$u_i^h = \sigma(u_i^{h-1}, AGG(u_j^{h-1}, j \in \mathbb{B}_i)), \quad (1)$$

where u_i^{h-1} is the representation obtained at the $(h-1)$ -th layer of node v_i , and u_i^0 is the node feature u_i of node v_i , \mathbb{B}_i represents the neighbors of node v_i , $AGG(\cdot)$ represents the aggregation function, σ represents the activation function such as *ReLU*.

2.2 Federated Graph Learning

A typical FGL system follows the FedAvg [McMahan *et al.*, 2017] algorithm. Specifically, the server sends an initial global model to all the clients. Then each client trains a local model with its local private data $G_i(V_i, \mathbb{E}_i)$ and shares its local model parameters with the server. The server then aggregates the local model parameters of all the clients to construct the global model's parameters, which can be formulated as:

$$W^t = \sum_{i=1}^N p_i W_i^t, \quad (2)$$

where N is the number of clients, W_i^t and p_i are trained model parameters at the t -th epoch and the weight of client i , respectively, W^t is the global model parameters at the t -th epoch. The optimization problem of FGL is formulated as:

$$W^* = \underset{W}{\operatorname{argmin}} \sum_{i=1}^N p_i \mathcal{L}(W, V_i, \mathbb{E}_i; Y_i), \quad (3)$$

where $\mathcal{L}(\cdot)$ represents the loss function, Y_i is node label in the node classification task or link label in the link prediction task of client i , respectively, V_i represents the set of nodes and \mathbb{E}_i represents the adjacency matrix of G_i .

2.3 Threat Model

As shown in Figure 1, we consider the FGL scenario in which the server is malicious. In this scenario, the server is not only interested in accessing the label distributions of the clients' private data, but also has the ability to manipulate the parameters of the deployed model. Furthermore, the server can analyze the gradients uploaded by the clients.

3 Related Work

The label distribution inference attack is a type of label recovery attack. [Zhu *et al.*, 2019] was the first to restore the training sample from gradients. They restored the input data and associated label from the gradients using gradient-matching. This method continuously optimizes the dummy input data and associated label by minimizing the mean square error of the gradients of the dummy sample with respect to the true gradients. [Zhao *et al.*, 2020] introduced iDLG and were the first to propose that with a non-negative activation function, privacy label can be extracted

Variance*1000 of I	41.052	23.719	21.283	10.640	4.547	2.031	0.817	0.224	0.067	0.003
Err	4.367	3.433	2.765	2.217	1.523	1.012	0.599	0.309	0.138	0.028

Table 1: Err with the variance of I when local epochs E is set to 1, the WikiCS dataset and a 2-layer GCN model are used.

with 100% success rate from the signs of gradients at the output layer. Both of these methods are applicable only to single-sample training scenarios. Afterwards, [Yin *et al.*, 2021; Geng *et al.*, 2021] extended the attacks to the mini-batch scenario, enhancing the applicability of the attacks.

Overall, most existing label recovery attacks focus on image datasets, and none of them discuss label distribution on graph datasets. In this paper, We focus on label distribution inference attacks against GNNs. More related work please refer Appendix A.

4 Message-Passing Issue in LDA

In this section, we will analyze how the message-passing mechanism in GNNs exacerbates LDA. We begin with an introduction to LDA. The prior analysis [Geng *et al.*, 2021] has shown that, for the k -th sample in a batch: $\Delta W_l^{(FC)}/E \approx \frac{1}{K} \sum_k (p_{k,l} \sum_m I_{k,m} - y_{k,l} \sum_m I_{k,m})$, where $\Delta W_l^{(FC)}$ is the sum of the gradients of the l -th output unit in the last fully connected layer along the input dimension, $I_{k,m}$ is the input of the k -th sample at the m -th input unit of the last fully connected layer, $p_{k,l}$ is the post-softmax probability at index l of the k -th sample, E is the local epochs. Furthermore, we have: $\sum_k y_{k,l} \cdot \bar{I} \approx \sum_k p_{k,l} I_k - K \Delta W_l^{(FC)}/E$, where $I_k = \sum_m I_{k,m}$, and \bar{I} is the mean value of I_k .

In order to separate out $\sum_k y_{k,l}$, we assume that I_k is close to its mean value, i.e., \bar{I} . Based on this assumption, we have:

$$d_l := \sum_k y_{k,l} \approx \frac{\sum_k p_{k,l} I_k}{\bar{I}} - \frac{K \Delta W_l^{(FC)}}{E \bar{I}}, \quad (4)$$

where d_l is the number of nodes inferred with label l ($l = 1, 2, \dots, L$). With the number of each label, we can compute the label distribution, which we define as D . In FL, we assume that the server can get E and $\Delta W_l^{(FC)}$ from clients. Additionally, the dummy training data randomly generated by the server can be used to estimate K , $p_{k,l}$, I_k , and \bar{I} . We named this method as LDA.

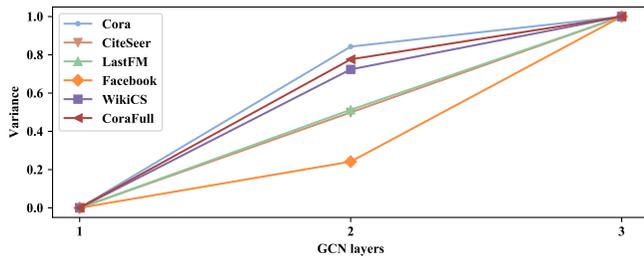


Figure 2: Illustrates the trend of variance of I with the number of GCN layers. The experimental results at this point are the normalized results.

Algorithm 1 Compression of node embeddings

Input: Global model parameters G , clipping threshold C
Output: The global model parameters G' after clipping and compression

```

1:  $\mathbb{N} \leftarrow 0, G' \leftarrow G$ 
2: for  $p \in G$  do
3:    $\mathbb{N} = \mathbb{N} + \|p\|_2^2$ 
4: end for
5:  $\mathbb{N} \leftarrow \mathbb{N}^{1/2}$ 
6: for  $p \in G, p' \in G'$  do
7:    $p' \leftarrow p / \max(1, \frac{\mathbb{N}}{C})$ 
8: end for
9: return  $G'$ 

```

Now we apply and analyze the LDA in GNNs. For the sake of description, we define $I := [I_1, I_2, \dots, I_K]$. Starting from the above assumption, i.e., I_k is close to \bar{I} , we can find that the error arises mainly because we replace I_k with \bar{I} , i.e., we consider that $\sum_k y_{k,l} I_k \approx \sum_k y_{k,l} \cdot \bar{I}$. We define the magnitude of the error as:

$$Err := \frac{1}{K} \sum_l \left| \sum_k y_{k,l} (I_k - \bar{I}) \right| = \frac{1}{K} \sum_k |I_k - \bar{I}|. \quad (5)$$

We can intuitively see that if the variance of I is smaller, the Err is smaller and the performance of the attack is better, and vice versa. To verify our conjecture, we conduct experiments on FGL by performing attacks during mid-training round and analyze the variance of I versus Err . Table 1 illustrates how Err varies with the variance of I , demonstrating that the Err declines as I 's variance declines, which confirms our suspicions.

As mentioned earlier, GNNs incorporate neighbor information through the message-passing mechanism. However, the message-passing mechanism also increases the variance of node embeddings. Moreover, the variance of the node embeddings increases with the number of GNN layers in the global model, and the node embeddings are directly related to I . We conduct experiments with GNN models featuring 1, 2, and 3 GCN layers and observe how the variance of I changes with the number of GCN layers. Figure 2 illustrates the variance of I as a function of the number of GCN layers. It can be seen that as the number of GCN layers increases, the variance of I becomes larger. Therefore, minimizing the variance of I is crucial for enhancing attack efficacy.

5 Our Methodology: EC-LDA

To minimize the variance of I , one effective approach is to constrain the absolute values of I_k within a small range, which can be achieved by clipping the model parameters.

Based on the above analysis, we now describe the design of EC-LDA, an enhanced attack for GNNs. Unlike LDA, EC-

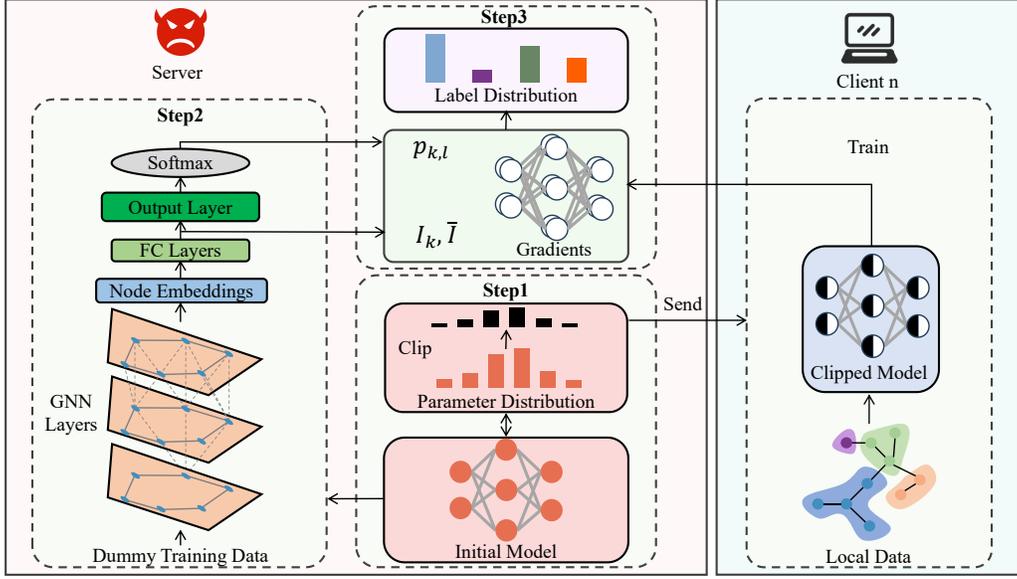


Figure 3: Illustration of the data flow during attack rounds in EC-LDA.

LDA makes full use of Equation 4 by clipping the parameters of the global model before distributing it to clients, which reduces the variance of node embeddings, thus making the variance of I smaller. As shown in Figure 3, EC-LDA consists of three main steps:

- **Step 1: Clipping model.** The server clips the initial model and sends it to clients to get the gradients. Algorithm 1 demonstrates the clipping method. First the ℓ_2 norm \mathbb{N} of the model is computed, and then $p/\max(1, \frac{\mathbb{N}}{C})$, i.e., the clipped parameters, will replace the original parameters p .
- **Step 2: Forward propagation of dummy data.** The server generates dummy training data randomly and inputs it into the initial model to obtain I_k, \bar{I} , and $p_{k,l}$.
- **Step 3: Calculating distribution.** After the server gets gradients, $p_{k,l}, I_k$, and \bar{I} , Equation 4 is used to infer the label distributions of clients' private data.

Algorithm 2 demonstrates EC-LDA in FGL. The boxed part represents the additional component of EC-LDA compared to the normal training process. First, the server initializes the model parameters and attack results. If the server does not perform an attack in the current round r , the training process is no different from the normal training process. If the server is going to perform an attack in round r , it will save the initial model W_{r-1} for round r as W'_{r-1} and clip W_{r-1} with Algorithm 1, i.e. **Step 1**. The server generates dummy training data and feeds it into W'_{r-1} to obtain $p_{k,l}, I_k$ and \bar{I} , i.e. **Step 2**. Then the server sends the clipped model down to all clients, and each client trains on its local private data and uploads the trained model. After training, server calculates the label distribution D with Equation 4, and saves D into \mathbb{L} , i.e. **Step 3**. The server uses the saved initial model W'_{r-1} of round r to replace W_r so that the model trained and aggregated on the clipped model can be avoided to be passed to

the next round. Therefore, a few attacks do not significantly impact model performance. The training process of clients does not differ from the general training process, clients first receive the model W_{r-1} from the server, then train the model W_{r-1} with the local privacy training data for E times and upload the trained model to the server. In other words, EC-LDA trades one round of training resources for one round of label distributions of clients' private training data.

Discussion. Implementing EC-LDA requires GNN layers with a fully connected final layer and the use of the cross-entropy loss function, which makes EC-LDA particularly versatile within classification tasks on GNNs, such as node classification. Since the link prediction is a binary classification task, EC-LDA is still applicable to link prediction tasks as long as the last fully connected layer of the model has two output units and the cross-entropy loss function is used. For node classification, EC-LDA can attack the label proportions, while for link prediction, it can target the graph density.

6 Experiments

In this section, we demonstrate the effectiveness of EC-LDA through answering the following three research questions:

- RQ1 - How effective is EC-LDA with real-world graph datasets, specifically for tasks such as node classification and link prediction?
- RQ2 - How robust is EC-LDA under different experimental variables?
- RQ3 - How the effectiveness of EC-LDA changes in the defense of differential privacy?

6.1 Experimental Settings

Here, we introduce evaluation metrics, model architecture, baselines, and relevant to the experiments as follow. The de-

Algorithm 2 EC-LDA against FGL (Server-side)

Input: Number of clients N , global rounds R , attack rounds A

Output: Final global model W_R , attack result \mathbb{L}

```
1: Initialize global model  $W_0$ , attack result  $\mathbb{L}$ 
2: for  $r = 1, 2, 3, \dots, R$  do
3:   if  $r \in A$  then
4:      $W'_{r-1} \leftarrow W_{r-1}$ , clipping  $W_{r-1}$  with Algorithm 1
5:     Generate dummy training data  $DTD$ 
6:     Input  $DTD$  into  $W'_{r-1}$  to obtain  $p_{k,l}, I_k$  and  $\bar{I}$ 
7:   end if
8:   for  $n = 1, 2, 3, \dots, N$  do
9:     Client  $n$  performs local training and uploads  $W_{r,n}$ 
       and gradients
10:   if  $r \in A$  then
11:     Calculate  $D$  based on Equation 4
12:     Add  $D$  to  $\mathbb{L}$ 
13:   end if
14: end for
15: Server aggregates all local models  $W_{r,n}$  to  $W_r$ 
16: if  $r \in A$  then
17:   Replace  $W_r$  with the initial model  $W'_{r-1}$ 
18: end if
19: end for
```

tails of datasets and hyper-parameter settings please refer to the Appendix E.

Evaluation Metrics: Inspired by [Geng, 2016], we use the following two evaluation metrics to fully demonstrate the effectiveness of EC-LDA: cosine similarity(cos-sim) and Jensen-Shannon divergence(JS-div). Cos-sim measures the similarity of two distributions and is applicable when the similarity of vectors is not directly related to the length of the vectors, while JS-div measures the distance between two distributions. Cos-sim takes values in the range $[-1, 1]$. The larger the cos-sim, the closer the inferred label distribution is to the ground-truth label distribution, i.e., the more effective the attack is. The value of JS-div ranges from $[0, 1]$, and the more effective the attack is, the smaller the JS-div is.

Model Architecture: To illustrate the general applicability of EC-LDA, we choose three classical GNN models as global models, which are GCN, GAT, and GraphSAGE. All of the above models consist of two parts, the GNN layers and the fully connected layers.

Baselines: We compare EC-LDA with three different attacks which are Infiltrator [Meng *et al.*, 2023], iLRG [Ma *et al.*, 2023], and LLG* [Wainakh *et al.*, 2021]. Infiltrator infers the label of the victim node by adding a neighbor to the victim node and observing the output of the neighbor. It is worth noting that Infiltrator focuses on node-level attacks, and for comparison, we attack all training nodes with Infiltrator. LLG* and iLRG both reveal the number of each label, and we use the number of samples per label extracted by iLRG and LLG* to compute the label distribution of each client. Similar to EC-LDA, iLRG and LLG* also use the gra-

dients of the last fully connected layer of the model, and when E is greater than 1, we use $\Delta W^{(FC)}/E$ as an approximation of the gradients.

Dummy Training Data: Based on the previous analysis, we know that the smaller the variance of I , the better the attack performance. Since the dummy training data is also related to the variance of I we generate it from a Gaussian distribution with a mean of 0 and a standard deviation of 0.001, containing 1000 nodes. This configuration enhances the attack performance.

6.2 Attack Performance (RQ1)

Node Classification. We evaluate the performance of EC-LDA with all datasets and all GNN types, which, for the node classification task, have a varying number of labels and are widely distributed, with 4 kinds of labels for nodes in the Facebook dataset and 70 kinds of labels for nodes in the Cora-Full dataset, the datasets with the smallest and the largest number of classes, respectively. Table 2 shows the experimental results. From the comparison presented in Table 2, it is evident that EC-LDA consistently demonstrates exceptional performance across various datasets and three distinct GNN models, as indicated by the cos-sim scores consistently at or above 0.999 and the JS-div consistently at or below 0.002, aligning closely with the optimal values. Remarkably, EC-LDA’s outstanding performance remains consistent regardless of the number of labels, the number of nodes, and the specific GNN types, showcasing its broad applicability. Additionally, EC-LDA consistently outperforms other methods across all experiments.

Link Prediction. In the link prediction experiments, we set the same number of positive and negative edges. Please refer to Appendix B for the experimental results. In all experiments of link prediction, cos-sim and JS-div reached 1.000 and 0.000, respectively, demonstrating the stunning performance of EC-LDA.

6.3 Ablation Experiments (RQ2)

To explore the effectiveness of EC-LDA under varying parameters, we performed extensive ablation experiments on the node classification tasks. Specifically, we explored the impact of the number of GNN layers, E , C , and number of clients separately. Please refer to the Appendix D for relevant experiments regarding the number of clients.

Impact of the Number of GNN Layers. We show the effect of the number of GNN layers on the performance of EC-LDA in Figure 4, where we use three different network structures with 1, 2, and 3 GNN layers, respectively, and each one is ubiquitous in GNNs. Figure 4 shows that as the GNN layers increase, EC-LDA performance improves, contrary to Figure 2. This is due to clipping reducing model parameters, which lowers the absolute values and variance of I_k , resulting in better attack performance. Overall, EC-LDA delivers excellent performance across three kinds of models.

Impact of C . In EC-LDA, the server will distribute the clipped model during attack rounds. The method of clipping directly affects the performance of EC-LDA. We employ the ℓ_2 norm clipping method and investigate the impact of C on EC-LDA. Figure 5 shows that the performance of

Dataset	GNN types	Cos-sim				JS-div			
		EC-LDA	Infiltrator	iLRG	LLG*	EC-LDA	Infiltrator	iLRG	LLG*
CoraFull	GCN	1.000	0.946	0.299	0.418	0.000	0.031	0.400	0.506
	GAT	1.000	0.977	0.314	0.395	0.000	0.017	0.415	0.515
	GraphSAGE	1.000	0.946	0.249	0.304	0.000	0.035	0.486	0.552
LastFM	GCN	1.000	0.953	0.590	0.414	0.000	0.040	0.212	0.410
	GAT	1.000	0.978	0.494	0.349	0.000	0.023	0.281	0.451
	GraphSAGE	1.000	0.966	0.470	0.347	0.000	0.038	0.315	0.448
WikiCS	GCN	1.000	0.865	0.610	0.475	0.001	0.058	0.212	0.402
	GAT	1.000	0.987	0.622	0.488	0.001	0.010	0.214	0.396
	GraphSAGE	1.000	0.721	0.561	0.512	0.001	0.130	0.242	0.374
Cora	GCN	1.000	0.981	0.601	0.558	0.001	0.009	0.267	0.327
	GAT	1.000	0.995	0.499	0.527	0.001	0.004	0.328	0.343
	GraphSAGE	1.000	0.987	0.469	0.588	0.000	0.009	0.382	0.312
CiteSeer	GCN	1.000	0.996	0.579	0.432	0.001	0.004	0.295	0.386
	GAT	1.000	0.997	0.522	0.439	0.001	0.004	0.294	0.383
	GraphSAGE	1.000	0.992	0.474	0.409	0.000	0.008	0.368	0.402
Facebook	GCN	1.000	0.991	0.679	0.581	0.000	0.003	0.195	0.326
	GAT	0.999	0.991	0.670	0.434	0.002	0.003	0.208	0.394
	GraphSAGE	1.000	0.982	0.643	0.519	0.000	0.008	0.251	0.354

Table 2: Performance of different attacks when training GCN, GAT, and GraphSAGE models on various datasets.

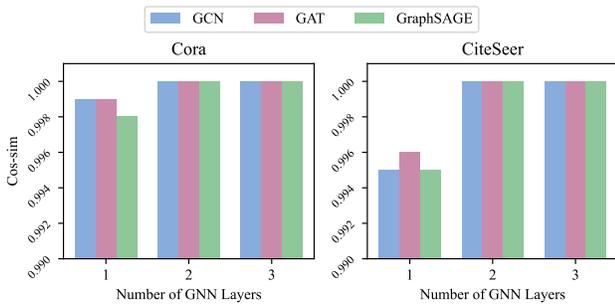


Figure 4: Performance of EC-LDA when the number of GNN layers of the model is 1, 2, and 3, respectively.

EC-LDA under different C . We can observe that as C increases, the performance of EC-LDA deteriorates. This is because with increasing C , the clipping intensity decreases. When C exceeds the ℓ_2 norm of the model parameters, the value of $\max(1, \frac{\| \cdot \|_2}{C})$ will be equal to 1, which means no clipping will be applied to the model, leading to a deterioration in EC-LDA’s performance.

Impact of E . To study the effect of different E on EC-LDA, we evaluate the performance of EC-LDA under different E . We show the experimental results in Table 3. It is evident that EC-LDA performs exceptionally well across varying E , with its performance across all datasets almost remaining unaffected by changes in E . As introduced in the previous section, we use $\Delta W_n^{(FC)}/E$ as an approximation of the gradients for EC-LDA. As E increases, the error between the true gradients and the approximated gradients be-

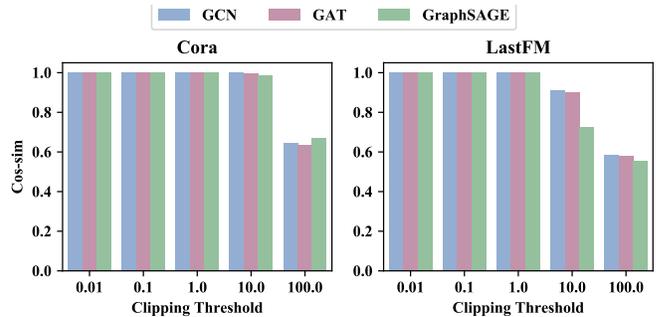


Figure 5: EC-LDA’s performance varies with changes in C .

comes larger, which results in a slight degradation of the performance of EC-LDA in some cases. Overall, EC-LDA performs effectively across different E .

6.4 Defense Performance of Differential Privacy (RQ3)

Differential privacy (DP) is a powerful privacy-preserving technology widely used in machine learning due to its rigorous mathematical definition. DP can defend against various attacks, such as membership inference attacks [Hui *et al.*, 2021], adversarial example attacks [Lecuyer *et al.*, 2019], and data reconstruction attacks [Balle *et al.*, 2022]. Specifically, we consider node-level differential privacy DP-GNN [Daigavane *et al.*, 2021] and label-level differential privacy Label-DP [Ghazi *et al.*, 2021]. DP-GNN controls the out-degree of nodes and is a variant of DP-SGD [Abadi *et al.*, 2016]. Label-DP adds noise to the label matrix to protect the labels of the training dataset.

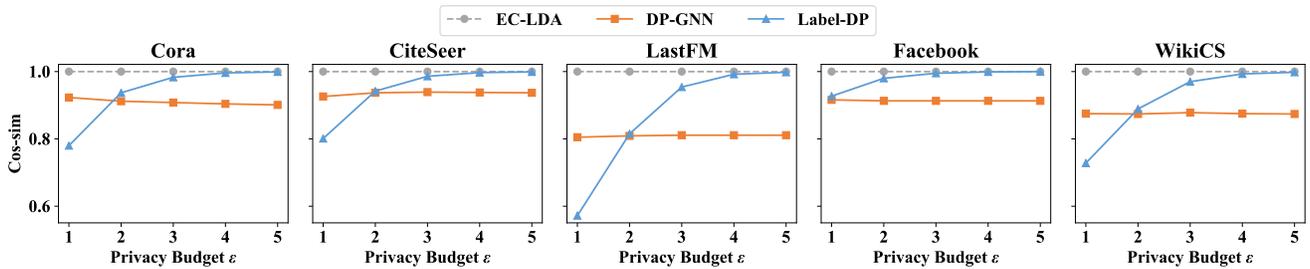


Figure 6: The attack performance in EC-LDA, DP-GNN and Label-DP.

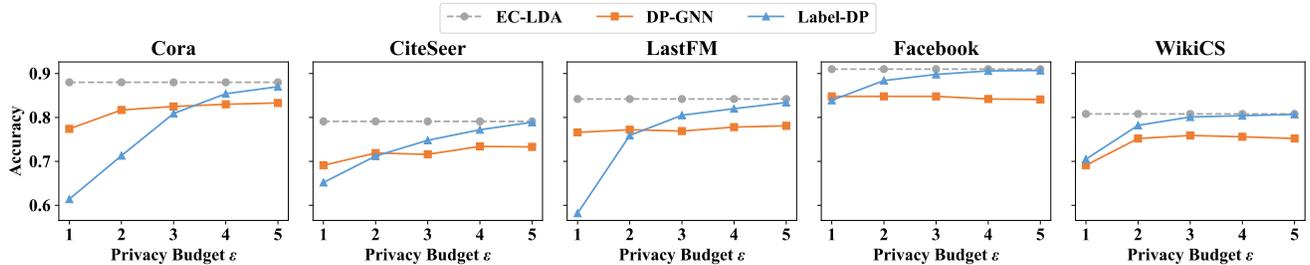


Figure 7: The model accuracy in EC-LDA, DP-GNN and Label-DP.

Figure 6 and Figure 7 show that the performance of EC-LDA and the final accuracy of the FGL model with privacy budget ϵ on node classification tasks under DP-GNN and Label-DP defenses, respectively. Privacy budget ϵ controls the magnitude of the added noise: a smaller privacy budget ϵ results in larger noise, and vice versa. In other words, a smaller privacy budget ϵ leads to a decline in the model’s final inference performance. From Figure 6, it is evident that the defense effectiveness of Label-DP improves with decreasing privacy budget ϵ . This improvement arises because Label-DP introduces increasing noise to the labels as privacy budget ϵ decreases. Additionally, the inferred label distribution closely matches the distribution after noise addition due to EC-LDA’s powerfulness in recovering label distributions. This alignment results in a significant disparity between the inferred distributions and actual label distributions, consequently diminishing EC-LDA’s attack effectiveness.

However, as privacy budget ϵ decreases, the defense effect of DP-GNN remains almost unchanged, which is because the fact that DP-GNN introduces Gaussian noise across all gradient layers, whereas in EC-LDA, the gradients of the last fully connected layer used for attacks are derived from summation along the input dimension. In the summation process, the added Gaussian noise will cancel out the positive and negative, reducing the overall noise, thereby leading to the insensitivity of the defense effect of DP-GNN to privacy budget ϵ . Therefore, the defense methods based on DP-SGD fail to be effective against EC-LDA.

7 Conclusion

This paper demonstrates the effectiveness of EC-LDA for label privacy in FGL scenarios. We discovered that LDA’s performance is influenced by the variance of node embeddings in GNNs and proposed EC-LDA, which improves attack perfor-

Model	Local Epochs	Cos-sim		JS-div	
		Cora	Facebook	Cora	Facebook
GCN	1	1.000	1.000	0.000	0.000
	3	1.000	1.000	0.000	0.000
	5	1.000	1.000	0.001	0.000
GAT	1	1.000	1.000	0.000	0.000
	3	1.000	1.000	0.000	0.001
	5	1.000	0.999	0.001	0.002
GraphSAGE	1	1.000	1.000	0.000	0.000
	3	1.000	1.000	0.000	0.000
	5	1.000	1.000	0.000	0.000

Table 3: Attack performance of EC-LDA when E is 1, 3, and 5, respectively.

mance by compressing node embeddings. Extensive experiments on six graph datasets show that EC-LDA outperforms SOTA methods in both node classification and link prediction tasks. We also explored the role of differential privacy in defending against EC-LDA. This paper primarily demonstrates the effectiveness of our attack method. In EC-LDA, the server clips the model’s parameters, actively disrupting the normal training process, and this behavior may be detected. Future work will focus on enhancing the stealthiness of attacks, potentially by training a fishing model with performance and parameter distributions similar to the original model for client deployment, to replace the compressed model.

References

- [Abadi *et al.*, 2016] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.
- [Aggarwal *et al.*, 2021] Abhinav Aggarwal, Shiva Kaviswanathan, Zekun Xu, Oluwaseyi Feyisetan, and Nathanael Teissier. Label inference attacks from log-loss scores. In *International Conference on Machine Learning*, pages 120–129. PMLR, 2021.
- [Balle *et al.*, 2022] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1138–1156. IEEE, 2022.
- [Bojchevski and Günnemann, 2017] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.
- [Chen *et al.*, 2023] Jinyin Chen, Minying Ma, Haonan Ma, Haibin Zheng, and Jian Zhang. An empirical evaluation of the data leakage in federated graph learning. *IEEE Transactions on Network Science and Engineering*, 2023.
- [Daigavane *et al.*, 2021] Ameeya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. Node-level differentially private graph neural networks. *arXiv preprint arXiv:2111.15521*, 2021.
- [Dang *et al.*, 2021] Trung Dang, Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, Peter Chin, and Françoise Beaufays. Revealing and protecting labels in distributed training. *Advances in neural information processing systems*, 34:1727–1738, 2021.
- [Fan *et al.*, 2019] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.
- [Geiping *et al.*, 2020] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.
- [Geng *et al.*, 2021] Jiahui Geng, Yongli Mou, Feifei Li, Qing Li, Oya Beyan, Stefan Decker, and Chunming Rong. Towards general deep leakage in federated learning. *arXiv preprint arXiv:2110.09074*, 2021.
- [Geng, 2016] Xin Geng. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1734–1748, 2016.
- [Ghazi *et al.*, 2021] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. Deep learning with label differential privacy. *Advances in neural information processing systems*, 34:27131–27145, 2021.
- [Gu and Bai, 2023] Yuhao Gu and Yuebin Bai. Ldia: Label distribution inference attack against federated learning in edge computing. *Journal of Information Security and Applications*, 74:103475, 2023.
- [Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [He *et al.*, 2020] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.
- [He *et al.*, 2021] 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.
- [Hui *et al.*, 2021] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. Practical blind membership inference attack via differential comparisons. *arXiv preprint arXiv:2101.01341*, 2021.
- [Jha *et al.*, 2022] Kanchan Jha, Sriparna Saha, and Hiteshi Singh. Prediction of protein–protein interaction using graph neural networks. *Scientific Reports*, 12(1):8360, 2022.
- [Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [Lecuyer *et al.*, 2019] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.
- [Ma *et al.*, 2023] Kailang Ma, Yu Sun, Jian Cui, Dawei Li, Zhenyu Guan, and Jianwei Liu. Instance-wise batch label restoration via gradients in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.
- [McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [Meng *et al.*, 2023] Lingshuo Meng, Yijie Bai, Yanjiao Chen, Yutong Hu, Wenyuan Xu, and Haiqin Weng. Devil in disguise: Breaching graph neural networks privacy through infiltration. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1153–1167, 2023.
- [Mernyei and Cangea, 2020] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

- [Qiu *et al.*, 2022] 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*, 20(5):3653–3668, 2022.
- [Rozemberczki and Sarkar, 2020] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1325–1334, 2020.
- [Rozemberczki *et al.*, 2021] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.
- [Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.
- [Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [Wainakh *et al.*, 2021] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User label leakage from gradients in federated learning. *arXiv preprint arXiv:2105.09369*, 2021.
- [Wang *et al.*, 2019] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Eavesdrop the composition proportion of training labels in federated learning. *arXiv preprint arXiv:1910.06044*, 2019.
- [Yin *et al.*, 2021] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16337–16346, 2021.
- [Zhang *et al.*, 2021] 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.
- [Zhao *et al.*, 2020] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.
- [Zhou *et al.*, 2022] Chunyi Zhou, Yansong Gao, Anmin Fu, Kai Chen, Zhiyang Dai, Zhi Zhang, Minhui Xue, and Yuqing Zhang. Ppa: Preference profiling attack against federated learning. *arXiv preprint arXiv:2202.04856*, 2022.
- [Zhu *et al.*, 2019] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A Related Work

Inference Attack against GNNs. [Chen *et al.*, 2023] investigated graph data leakage in horizontal federated and vertical federated scenarios. Specifically, link inference attack and attribute inference attack are proposed in the vertical federated scenario, graph reconstruction attack and graph feature attack are proposed in the horizontal federated scenario. [Qiu *et al.*, 2022] proposed an inference attack against the relationships between nodes of GNNs in vertical federated scenarios. [Meng *et al.*, 2023] inferred the labels of victim nodes by infiltrating the raw graph data, i.e., by adding elaborately designed nodes and edges, and their attacks can only be performed on trained models. However, their attacks focus on the node-level, and there are no extant papers on label distribution inference attacks at the graph-level for FGL.

Label Recovery Attack. The label distribution inference attack is a type of label recovery attack. [Zhu *et al.*, 2019] was the first to restore the training sample from gradients. They restored the input data and associated label from the gradients using gradient-matching. This method continuously optimizes the dummy input data and associated label by minimizing the mean square error of the gradients of the dummy sample with respect to the true gradients. [Zhao *et al.*, 2020] introduced iDLG and were the first to propose that with a non-negative activation function, privacy label can be extracted with 100% success rate from the signs of gradients at the output layer. Both of these methods are applicable only to single-sample training scenarios. Afterwards, [Yin *et al.*, 2021; Geng *et al.*, 2021] extended the attacks to the mini-batch scenario. Among them, [Yin *et al.*, 2021] also requires the use of non-negative activation functions and does not allow duplicate labels within a mini-batch. Therefore, their method is applicable when the number of classes in the dataset is far greater than the batch size. [Gu and Bai, 2023] believes that the label distribution of the training dataset will leave a footprint in the parameter changes of the model output layer. They use the auxiliary dataset to obtain the connection between the gradients of the output layer and the label distribution of the training data, then train a neural network model accordingly to obtain the label distribution information of the training dataset. [Aggarwal *et al.*, 2021] relies on number theory and combinatorics to recover label information from log-loss scores. However, their method is sensitive to the number of classes in the dataset, and as the variety of labels increases, not only does the accuracy decrease, but the inference time also increases. [Ma *et al.*, 2023] recovered class-wise embeddings from the gradients and further restored the number of each label. [Dang *et al.*, 2021] proposes RLG, which extracts label information with the gradients of the output layer, but RLG requires the use of the soft-max activation function. Additionally, RLG only reveals which labels are used for training, without disclosing the quantity of samples per label. [Zhou *et al.*, 2022] proposed PPA, which can infer a client’s label information but can only output the majority class or minority class. It cannot fully reflect the label information. [Wainakh *et al.*, 2021] proposed LLG, which utilizes the magnitude and direction of shared gradients to determine whether a specific label is present. LLG has three versions,

suitable for different scenarios. Among them, LLG assumes that the attacker can only access shared gradients, LLG* assumes that the attacker can access both the model’s parameters and gradients, and LLG+ assumes that the attacker has an auxiliary dataset. [Wang *et al.*, 2019] proposed three attack methods, which can infer whether a specific label appears in the training process, the quantity of each label for a specific client in a round, and the quantity of each label throughout the entire training process, respectively.

Overall, most existing label recovery attacks focus on image datasets, and none of them discuss label distribution on graph datasets. In this paper, We focus on label distribution inference attack against GNNs.

B Experimental Results of the Link Prediction Tasks

Table 4 shows the performance of EC-LDA on all datasets in the link prediction task, where we can see that EC-LDA achieves the optimal values on both metrics, demonstrating an impressive performance.

Dataset	GNN types	Cos-sim	JS-div
Facebook	GCN	1.000	0.000
	GAT	1.000	0.000
	GraphSAGE	1.000	0.000
CiteSeer	GCN	1.000	0.000
	GAT	1.000	0.000
	GraphSAGE	1.000	0.000
Cora	GCN	1.000	0.000
	GAT	1.000	0.000
	GraphSAGE	1.000	0.000
WikiCS	GCN	1.000	0.000
	GAT	1.000	0.000
	GraphSAGE	1.000	0.000
LastFM	GCN	1.000	0.000
	GAT	1.000	0.000
	GraphSAGE	1.000	0.000
CoraFull	GCN	1.000	0.000
	GAT	1.000	0.000
	GraphSAGE	1.000	0.000

Table 4: Performance of EC-LDA when training GCN, GAT, and GraphSAGE models on various datasets for link prediction tasks.

C Label Distribution

We employ a community detection algorithm to partition the graph dataset for each client, which identifies communities within the network(i.e., groups of nodes with high connection densities). Figure 8 shows the distribution of labels assigned to each client by this algorithm for the Cora dataset. From the figure, it is evident that, with a setup of 10 clients, the algorithm simulates a variety of label distributions. Our prior

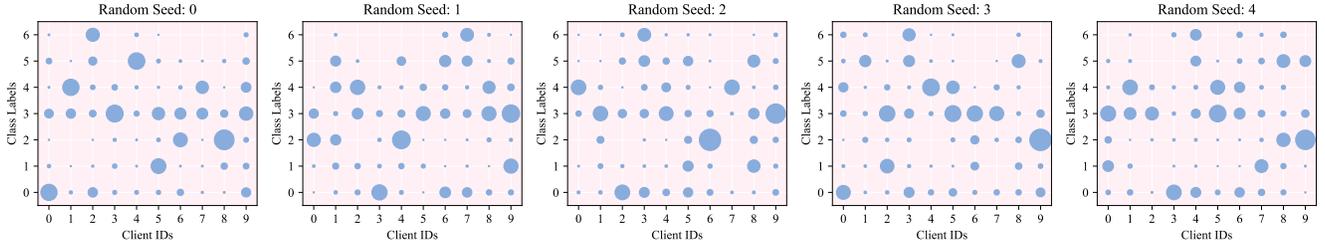


Figure 8: The label distribution obtained by each client when dividing the Cora dataset under five different random seeds.

experimental results indicate that EC-LDA consistently delivers strong performance. Therefore, we conclude that EC-LDA can perform well across a range of distributions.

D Impact of Number of Clients

To investigate the impact of the number of clients on the performance of EC-LDA, we conducted experiments on the Cora dataset, varying the number of clients from 5 to 50. Figure 9 demonstrates the performance of EC-LDA as the number of clients changes. We can see that regardless of the number of clients, the performance of EC-LDA remains excellent, showcasing its wide applicability.

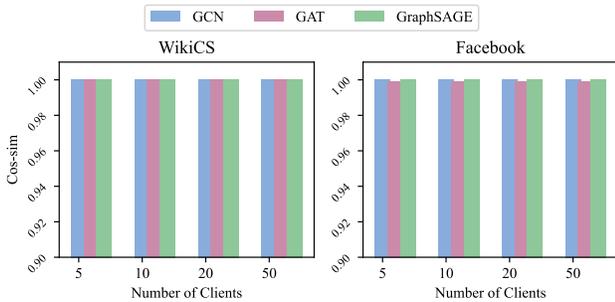


Figure 9: The performance of EC-LDA when the number of clients is 5, 10, 20, and 50, using the WikiCS and Facebook datasets.

E Datasets and Hyper-parameter Settings

We conduct experiments on six widely used public datasets: Cora [Sen *et al.*, 2008], CiteSeer [Sen *et al.*, 2008], LastFM [Rozemberczki and Sarkar, 2020], Facebook [Rozemberczki *et al.*, 2021], CoraFull [Bojchevski and Günnemann, 2017], and WikiCS [Mernyei and Cangea, 2020]. Cora, CiteSeer, WikiCS, and CoraFull are citation network datasets. Facebook and LastFM are social network datasets. We show the main features of these datasets in Table 5. To demonstrate the effectiveness of EC-LDA under conditions with a large number of labels, we select the CoraFull dataset, which has up to 70 labels.

All datasets use the SGD optimizer. If not specified, all the experiments in this paper use the following setup: all at E of 5, clipping threshold C of 0.01, number of clients of 10, and

Network Types	Datasets	#Nodes	#Edges	#Features	#Classes
Citation Network	CiteSeer	2120	7358	3703	6
	Cora	2485	10138	1433	7
	WikiCS	11311	297033	300	10
Social Network	CoraFull	18800	125370	8710	70
	Facebook	22470	342004	128	4
	LastFM	7624	55612	128	18

Table 5: Main features of the datasets

attack on all the clients in the middle round. All the experiments in this paper are taken with different random seeds to repeat 5 times, and the results are averaged.