Vertical Federated Graph Neural Network for Recommender System

Peihua Mai [1] **Yan Pang** [1]

Abstract

Conventional recommender systems are required to train the recommendation model using a centralized database. However, due to data privacy concerns, this is often impractical when multi-parties are involved in recommender system training. Federated learning appears as an excellent solution to the data isolation and privacy problem. Recently, Graph neural network (GNN) is becoming a promising approach for federated recommender systems. However, a key challenge is to conduct embedding propagation while preserving the privacy of the graph structure. Few studies have been conducted on the federated GNN-based recommender system. Our study proposes the first vertical federated GNN-based recommender system, called VerFedGNN. We design a framework to transmit: (i) the summation of neighbor embeddings using random projection, and (ii) gradients of public parameter perturbed by ternary quantization mechanism. Empirical studies show that VerFedGNN has competitive prediction accuracy with existing privacy preserving GNN frameworks while enhanced privacy protection for users' interaction information.

1. Introduction

Graph neural network (GNN) has become a new state-of-art approach for recommender systems. The core idea behind GNN is an information propagation mechanism, i.e., to iteratively aggregate feature information from neighbors in graphs. The neighborhood aggregation mechanism enables GNN to model the correlation among users, items, and related features. Compared with traditional supervised learning algorithms, GNN can model high-order connectivity through multiple layers of embedding propagation and thus capture the similarity of interacted users and items (Gao et al., 2022). Besides, the GNN-based model can effectively alleviate the problem of data sparsity by encoding semi-supervised signals over the graph (Jin et al., 2020). Benefiting from the above features, GNN-based models have shown promising results and outperformed previous methods on the public benchmark datasets (Berg et al., 2017; Wang et al., 2019b;a).

Instead of training the model based on their own graphs, organizations could significantly improve the performance of their recommendation by sharing their user-item interaction data. However, such sharing might be restricted by commercial competition or privacy concerns, leading to the problem of data isolation. For example, the data protection agreement might prohibit the vendors from transferring users' personal data, including their purchase and clicking records, to a third party.

Federated learning (FL), a machine learning setting with data distributed in multiple clients, is a potential solution to the data isolation problem. It enables multiple parties to collaboratively train a model without sharing their local data (Peyré et al., 2019). A challenge in designing the federated GNN-based recommender system is how to perform neighborhood aggregation while keeping the graph topology information private. Indeed, each client should obtain the items their users have interacted with in other organizations to conduct embedding propagation. However, the user interaction data are supposed to keep confidential in each party, adding to the difficulty of federated implementation.

Few studies have been conducted on the federated implementation of GNN-based recommender systems. To our best knowledge, FedPerGNN is the first federated GNN-based recommender system on user-item graphs (Wu et al., 2022a). However, their work considers a horizontal federated setting where each client shares the same items but with different users. This paper studies the vertical federated setting in which multiple collaborating recommenders offer different items to the same set of users. In addition, FedPerGNN expands the local user-item graphs with anonymous neighbouring user nodes to perform embedding propagation, which could leak users' interaction information (see Section 4.1).

To fill the gap, this paper proposes a vertical federated learning framework for the GNN-based recommender system

[1]Department of Analytics and Operations, National University of Singapore, 119077 Singapore. Correspondence to: Yan Pang <jamespang@nus.edu.sg>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

(VerFedGNN)[1]. Our method transmits (i) the neighbor embedding aggregation reduced by random projection, and (ii) gradients of public parameter perturbed by ternary quantization mechanism. The privacy analysis suggests that our approach could protect users' interaction data while leveraging the relation information from different parties. The empirical analysis demonstrates that VerFedGNN significantly enhance privacy protection for cross-graph interaction compared with existing privacy preserving GNN frameworks while maintaining competitive prediction accuracy.

Our main contributions involves the following:

- To the best of our knowledge, we are the first to study the GNN-based recommender system in vertical federated learning. We also provide a rigorous theoretical analysis of the privacy protection and communication cost. The experiment results show that the performance of our proposed federated algorithm is comparable to that in a centralized setting.
- We design a method to communicate projected neighborhood aggregation and quantization-based gradients for embedding propagation across subgraphs. Our method outperforms existing framework in terms of accuracy and privacy. The proposed framework could be generalized to the horizontal federated setting.
- We propose a de-anonymization attack against existing federated GNN framework that infers cross-graph interaction data. The attack is simulated to evaluate its performance against different privacy preserving GNN approaces.

2. Literature Review

Graph Neural Network (GNN) for Recommendation: There has been a surge of studies on designing GNN-based recommender systems in recent years. Graph convolutional matrix completion (GC-MC) employs a graph auto-encoder to model the direct interaction between users and items (Berg et al., 2017). To model high-order user-item connectivities, neural graph collaborative filtering (NGCF) stacks multiple embedding propagation layers that aggregate embeddings from neighbouring nodes (Wang et al., 2019b). LightGCN simplifies the design of NGCF by demonstrating that two operations, feature transformation and nonlinear activation, are redundant in the GCN model (He et al., 2020). PinSage develops an industrial application of GCN-based recommender systems for Pinterest image recommendation (Ying et al., 2018). To achieve high scalability, it samples a neighborhood of nodes based on a random walk and performs localized aggregation for the node.

[1]Source code: https://github.com/maiph123/VerticalGNN

Federated Graph Neural Network: Recent research has made progress in federated graph neural network. Most of the work either performs neighborhood aggregation individually (Zhou et al., 2020; Liu et al., 2021), or assumes that the graph topology could be shared to other parties (Chen et al., 2021). It's a non-trivial task to incorporate the cross-client connection while preserving the relation information between nodes. To protect the graph topology information, a direct method is to apply differential privacy on the adjacency matrix (Zhang et al., 2021a), which requires a tradeoff between privacy protection and model performance. In FedSage+, a missing neighbour generator is trained to mend the local subgraph with generated cross-subgraph neighbors (Zhang et al., 2021b). To our best knowledge, FedPerGNN is the first work that develops horizontal federated GNN-based recommender system with user-item graphs (Wu et al., 2022a). Each client expands the local user-item graphs with anonymous neighbouring user node to learn high-order interaction information.

This paper considers the unexplored area, i.e., graph learning for recommender systems in a vertical federated setting. We show that applying local graph expansion could leak user interaction information from other parties, and develop a GNN-based recommender system that leverages the neighbourhood information across different subgraphs in a privacy-preserving way.

3. Problem Formulation and Background

3.1. Problem Statement

We assume that P parties collaboratively train a recommender system, where each party holds a set of common users but non-overlapping items. Denote $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ as the set of common users and $\mathcal{V}_p = \{v_1, v_2, ..., v_{M_p}\}$ as the set of items for party p. The total item size M_p, $p \in [1, P]$ is shared across parties. Assume that user u has N_u neighboring items $\mathcal{N}(u) = \{v_1, v_2, ..., v_{N_u}\}$, and item v has N_v neighboring users $\mathcal{N}(v) = \{u_1, u_2, ..., u_{N_v}\}$. Denote $\mathcal{N}_p(u) = \{v_1, v_2, ..., v_{N_u^p}\}$ as the neighboring items for user u in party p. The related items and users form a local subgraph $\mathcal{G}_p$ in party p. Denote r_{uv} as the rating user u gives to item v. Our objective is to generate a rating prediction that minimizes the squared discrepancy between actual ratings and estimate.

3.2. Graph Neural Network (GNN)

General Framework: We adopt a general GNN (Wu et al., 2022b) framework as the underlying model for our recommender system. In the initial step, each user and item is offered an ID embedding of size D, denoted by $e_u^0, e_v^0 \in R^D$ respectively. The embeddings are passed through K mes-

sage propagation layers:

$$\begin{aligned} n_u^k &= Agg_k(\{e_v^k, \forall v \in \mathcal{N}(u)\}) \\ e_u^{k+1} &= Update_k(e_u^k, n_u^k) \end{aligned} \tag{1}$$

$$\begin{aligned} n_v^k &= Agg_k(\{e_u^k, \forall u \in \mathcal{N}(v)\}) \\ e_v^{k+1} &= Update_k(e_v^k, n_v^k) \end{aligned} \tag{2}$$

where e_u^k and e_v^k denote the embeddings at the kth layer for user u and item v, and Agg_k and $Update_k$ represent the aggregation and update operations, respectively. The final representation for users (items) are given by the combination of embeddings at each layer:

$$h_u = \sum_{k=0}^{K} a_k e_u^k; \; h_v = \sum_{k=0}^{K} a_k e_v^k \tag{3}$$

where h_u and h_v denote the final representation for user u and item v respectively, K denotes the number of embedding propagation layers, and a_k denotes the trainable parameter implying the weight of each layer.

The rating prediction is defined as the inner product of user and item representation

$$\hat{r}_{uv} = h_u^T h_v \tag{4}$$

The loss function is computed as:

$$\mathcal{L} = \sum_{(u,v)} (\hat{r}_{uv} - r_{uv})^2 + \frac{1}{N} \sum_u \|e_u^0\|_2^2 + \frac{1}{M} \sum_v \|e_v^0\|_2^2 \tag{5}$$

where (u, v) denotes pair of interacted user and item, and N and M denote the number of users and items respectively.

Aggregation and Update Operations: This paper discusses three typical GNN frameworks: Graph convolutional network (GCN) (Kipf & Welling, 2016) , graph attention networks (GAT) (Veličković et al., 2017), and gated graph neural network (GGNN) (Li et al., 2015). We illustrate their aggregation and update operations for user embedding as an example.

- **GCN** approximates the eigendecomposition of graph Laplacian with layer-wise propagation rule:

$$\begin{aligned} Agg_k: \; n_u^k &= \sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{N_u N_v}} e_v^k \\ Update_k: \; e_u^{k+1} &= \sigma(W^k(e_u^k + n_u^k)) \end{aligned} \tag{6}$$

- **GAT** leverages self-attention layers to assign different importances to neighboring nodes:

$$\begin{aligned} Agg_k: \; n_u^k &= \sum_{v \in \mathcal{N}(u)} b_{uv}^k e_v^k \\ Update_k: \; e_u^{k+1} &= \sigma(W^k(b_{uu}^k e_u^k + n_u^k)) \end{aligned} \tag{7}$$

where b_{uu}^k and b_{uv}^k are the importance coefficients computed by the attention mechanism:

$$b_{uv}^k = \frac{exp(\tilde{b}_{uv}^k)}{\sum_{v' \in \mathcal{N}(u) \cup u} exp(\tilde{b}_{uv'}^k)} \tag{8}$$

where $b_{uv}^k = Att(e_u^k, e_v^k)$ is computed by an attention function.

- **GGNN** updates the embeddings with a gated recurrent unit (GRU):

$$\begin{aligned} Agg_k: \; n_u^k &= \sum_{v \in \mathcal{N}(u)} \frac{1}{N_u} e_v^k \\ Update_k: \; e_u^{k+1} &= GRU(e_u^k, n_u^k) \end{aligned} \tag{9}$$

3.3. Differential Privacy

We adopt the standard definition of (ϵ, δ)-differential privacy (Dwork et al., 2014) for our analysis.

Definition 3.1. A randomized function $M(x)$ is (ϵ, δ)-differentially private if for all x, y such that $\|x - y\|_1 \leq 1$ and any measurable subset $S \subseteq \text{Range}(M)$,

$$P(M(x) \in S) \leq e^{\epsilon} P(M(y) \in S) + \delta \tag{10}$$

This paper assumes an untrusted server and requires that the local gradients from each party satisfy (ϵ, δ)-local differential privacy (Duchi et al., 2013).

4. Proposed Method

4.1. De-anonymization Attack

A straightforward cross-graph propagation solution is to use anonymous neighborhood embeddings from other graphs (Wu et al., 2022a). Adapting to the vertical setting, each party sends the encrypted ids of each item's related users to the server, and the server provides each party with the embeddings of their neighboring items via user matching.

One privacy concern of this approach suffers leakage of interaction information, as is shown by the de-anonymization attack below. We assume that each organization is accessible to the set of available items from other parties.

Suppose that party A wants to obtain their users' interaction with party B. Party A could create a set of adversarial users that have registered on other platforms. Each fake user rates only one item in party B. The interaction information could be recovered by matching the embeddings for adversarial and honest users. Denote $v_i \in \mathcal{N}(u)$ as the ith item in user u's neighborhood from party B, $\mathcal{N}_{adv}$ as the set of items given by all fake users, and $v_j \in \mathcal{N}_{adv}$ as the item rated by j^{th} adversarial user. The inferred i^{th} item for user u is:

$$\hat{v}_i = \arg\min_{v' \in \mathcal{N}_{adv}} \|e'^0_v - e^0_{v_i}\|_1 \tag{11}$$

where $\|\cdot\|$ computes the l_1-norm of inner vectors.

The above attack suggest that revealing the individual embedding for each item is susceptible to privacy leakage. Following, we introduce a new method to obtain embeddings at the aggregated level.

4.2. Federated Graph Neural Network

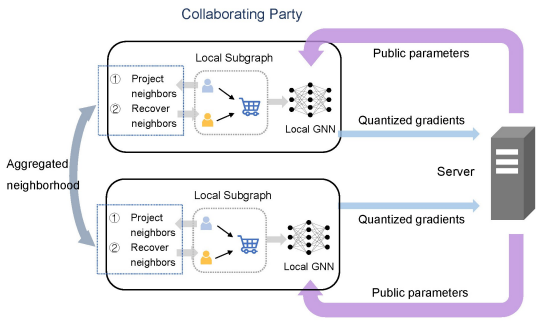


Figure 1. Overall framework of VerFedGNN

In the proposed framework, the collaborating parties jointly conduct model training with a central server. Each client cl_p is associated with a party p. Item embeddings e^0_v should be maintained privately on each client, while other public parameters are initialized and stored at the server. At the initial step, Private Set Intersection (PSI) is adopted to conduct secure user alignment (Pinkas et al., 2014). Algorithm 1 outlines the process to perform VerFedGNN. Figure 1 gives the overall framework of our VerFedGNN, and we will illustrate the key components in the following.

4.3. Neighborhood Aggregation

Instead of sending the individual embeddings of missing neighbors, each party performs embedding aggregation locally for each common user before the transmission. Each party outputs a list of $N \times D$ aggregation matrices $[X^0_p, X^1_p, ..., X^{K-1}_p]$, with each row of X^k_p given by $E_p(n^k_u)$. Below details the local neighborhood aggregation for the three GNN frameworks:

GCN requires N_u to perform local aggregation, while sharing N_u could reveal how many items user u has interacted with in other parties. To preserve the privacy of N^p_u, we develop an estimator from party p's view in replacement of N_u:

$$E_p(N_u) = \frac{\sum_i M_i}{M_p} \cdot N^p_u \tag{12}$$

Algorithm 1 Federated Vertical Graph Neural Network

FL Server:
Initialize public parameters.
Initialize projection matrix Φ and broadcast the seed.
for $t \in [1, T]$ **do**
  Distribute public parameters to clients $p \in \mathcal{A}_t$
  Receive and aggregate local gradients from client p for $p \in \mathcal{A}_t$
  Update public parameters with aggregated gradients
end for

Client $c, c \in [1, P]$**:**
Initialize item embeddings.
for $t \in [1, T]$ **do**
  Download public parameters from server
  Received projected embedding aggregation Y^k_p for layer $k \in [0, K-1]$ from parties $p \in \mathcal{A}_t \backslash c$
  Compute aggregated embeddings matrix X^k_c for layer $k \in [0, K-1]$
  Derive projected matrix Y^k_c using expression 17, and send to parties $p \in \mathcal{A}_t \backslash c$
  Received projected matrix Y^k_p for layer $k \in [0, K-1]$ from parties $p \in \mathcal{A}_t \backslash c$
  Reconstruct aggregated embeddings matrix using expression 18 for layer $k \in [0, K-1]$ and parties $p \in \mathcal{A}_t \backslash c$
  Conduct layer-wise embedding update with $\hat{X}^k_p$ for $k \in [0, K-1]$ and $p \in \mathcal{A}_t \backslash c$
  Calculate gradients locally and update private parameters e^0_v for $v \in \mathcal{V}_p$
  Perturb gradients for public parameters using ternary quantization scheme given by Definition 4.2
  Upload quantized gradients to server
end for

where M_i denotes the number of items in party i. The estimator is utilized to perform embedding aggregation:

$$E_p(n^k_u) = \sum_{v \in \mathcal{N}_p(u)} \frac{1}{\sqrt{E_p(N_u)N_v}} e^k_v \tag{13}$$

GAT calculates importance coefficient b^k_{uv} using all item embeddings for $v \in \mathcal{N}_u$, incurring further privacy concern and communication cost. Therefore, we adapt equation 8 to obtain $E_p(b^k_{uv})$:

$$E_p(b^k_{uv}) = \frac{exp(\tilde{b}^k_{uv})}{exp(\tilde{b}^k_{uu} + \sum_{v' \in \mathcal{N}_u} exp(\tilde{b}^k_{uv'}) \cdot \sum_i M_i / M_p)} \tag{14}$$

The neighbor items are aggregated locally using:

$$E_p(n^k_u) = \sum_{v \in \mathcal{N}_p(u)} b^k_{uv} e^k_v \tag{15}$$

GGNN slightly adapt Agg_k in equation 9 to perform aggregation:

$$E_p(n_u^k) = \sum_{v \in \mathcal{N}_p(u)} \frac{1}{N_u^p} e_v^k \tag{16}$$

Refer to Appendix A for embedding update with the aggregated neighborhood.

4.4. Random Projection

Though neighborhood aggregation reduces the information leakage, users might still be susceptible to de-anonymization attack when they rated few items in other parties. We adopt random projection (Lindenstrauss, 1984) to perform multiplicative data perturbation for two reasons: (1) random projection allows to reduce dimensionality and reconstruct matrix without prior knowledge of the data; (2) random projection preserve the pairwise distance between points with small error (Ghojogh et al., 2021). Below we define a Gaussian random projection matrix.

Definition 4.1. For $q \ll N_u$, a Gaussian random projection matrix $\Phi \in \mathbb{R}^{q \times N_u}$ has elements drawn independently from Gaussian distribution with mean 0 and variance $1/q$.

Each active party sends a list of $q \times D$ projected matrices to other participants:

$$Y_p^k = \Phi X_p^k \tag{17}$$

for $k \in [0, K-1]$. The recipient recover the perturbed aggregation matrices $\hat{X}_p^k,\ k \in [0, K-1]$:

$$\hat{X}_p^k = \Phi^T Y_p^k \tag{18}$$

4.5. Privacy-preserving Parameter Update

The gradients of public parameters could leak sensitive information about the users. For example, if two users rated the same items, the gradients for their embeddings would be similar. Therefore, a participant could infer subscribers' interaction history by comparing their embeddings with adversarial users. We introduce a ternary quantization scheme (Wang & Başar, 2022) to address this issue.

Definition 4.2. The ternary quantization scheme quantizes a vector $x = [x_1, x_2, ..., x_d]^T \in \mathbb{R}^d$ as follows:

$$Q(x) = [q_1, q_2, ..., q_d],\ \ q_i = r\text{sign}(x_i)b_i,\ \ \forall 1 \le i \le d \tag{19}$$

where r is a parameter such that $\|x\|_\infty \le r$, sign represents the sign of a value, and b_i are independent variables following the distribution

$$\begin{cases} P(b_i = 1|x) = |x_i|/r \\ P(b_i = 0|x) = 1 - |x_i|/r \end{cases} \tag{20}$$

The ternary quantization scheme is adopted in place of Gaussian or Laplace noise for two reasons: (1) The scale of Gaussian or Laplace noise is determined by the sensitivity that could be significant for high dimensional data. For GNN model with large size, directly adding the calibrated noises would greatly distort the direction of gradients. On the other hand, the quantization scheme ensures that the sign of each element in the gradient is not reversed by the stochastic mechanism. (2) For user embeddings, the gradients is a $N_u \times D$ matrix with communication cost proportional to user size. Under the quantization scheme, parties could send a much smaller sparse matrix indexed by the non-zero binary elements.

4.6. Partial Participants

We consider the case where in each iteration, only a portion of clients participates in model training. The challenge is that both the embedding update and gradient computation contain components summed over all clients to capture the complete graph structure. To address this issue, we develop an estimator of the total summation based on the subsets of components received.

Denote c_i as the component send by party i, and $\mathcal{A}_t$ as the set of participating clients in iteration t. The estimation $E(C)$ is given by:

$$E(C) = \frac{\sum_i M_i}{\sum_{i \in \mathcal{A}_t} M_i} \sum_i c_i \tag{21}$$

Specifically, the component c_i is $E_p(n_u^k),\ k \in [0, K-1]$ for embedding update, and local gradients for gradient computation, respectively.

5. Theoretical Performance Analysis

5.1. Privacy Analysis

The privacy of the algorithm is analyzed for two communication stages: (i) neighborhood exchange, and (ii) gradient transmission. We assume honest-but-curious (Yang et al., 2019) participants for the analysis, i.e., the participant will not deviate from the defined training protocol but attempt to learn information from legitimately received messages.

Neighborhood exchange: Suppose that an attacker would like to infer the original aggregation matrix X_p^k given Y_p^k. The model can be analyzed as an underdetermined system of linear equations with more unknowns than equations $y = \Phi x$, where x is a column vector in X_p^k and y is the corresponding column in Y_p^k. We start with the definition of l-secure (Du et al., 2004).

Definition 5.1. A matrix Φ is l-secure if a submatrix Φ_k formed by removing any l columns from Φ has full row rank.

Lemma 5.2. *Let Ψ be an $l \times N$ matrix, where each row is a nonzero linear combination of row vectors in Φ. If Φ l-secure, the linear equations system $y = \Psi x$ involves at least $2l$ variables if these l vectors are linearly independent.*

Theorem 5.3. *For $2q \leq m+1$, let Φ be a $q \times m$ matrix with entries independently chosen from Gaussian distribution. For a linear system of equations $y = \Phi x$, it's impossible to solve the exact value of any element in x.*

The proof is given in Appendix B and C. As long as we select $q \leq (m+1)/2$, the privacy of aggregation matrix is protected in the sense that the attacker cannot identify the exact value of any elements in the original data.

Next, we consider the possibility to infer users' interaction history from the reconstructed aggregation matrix. Appendix E demonstrates the NP-hardness of finding a subset of items that match the aggregated embeddings.

Gradient transmission: The gradient transmitted to server is perturbed by the ternary quantization scheme. The following theorem shows that the ternary quantization can achieve $(0, \frac{1}{r})$ differential privacy.

Theorem 5.4. *The ternary quantization scheme given by Definition 4.2 achieves $(0, \frac{1}{r})$-differential privacy for individual party's gradients in every iteration.*

Proof. The privacy guarantee has been proved by (Wang & Başar, 2022) in Theorem 3. □

Remark 5.5. The ternary quantization still achieves $(0, \frac{1}{r})$-differential privacy when the l_1 norm in Definition 3.1 is replaced with any l_p norm with $p \geq 1$ (see Appendix D).

5.2. Utility Analysis

The federated algorithm involves two sources of error: (i) random projection and reconstruction of aggregation matrix, and (ii) stochastic ternary quantization mechanism. We will discuss the concerns one at a time.

Random projection and matrix reconstruction: The reconstructed matrix $X_p^k,\ k \in [0, K-1],\ p \in [1, P-1]$ doesn't deviate much from the original matrix with the following bounded MSE.

Theorem 5.6. *Let Φ be a random matrix defined in Definition 4.1. For any $X \in \mathbb{R}^{N_u \times D}$,*

$$\mathbb{E}_\Phi[\|\Phi^T \Phi X - X\|_F^2] = \frac{(m+1)}{p}\|X\|_F^2 \tag{22}$$

where $\|\cdot\|$ represents the Frobenius norm of inner matrix.

Refer to Appendix F for the proof of Theorem 5.6.

Ternary quantization mechanism: In Appendix G we provide an convergence analysis for the gradient perturbed mechanism.

5.3. Communication Analysis

The communication cost is analyzed in terms of the total message size transferred between parties. We assume that the participation rate α, the number of participating clients divided by that of total clients in an iteration, remains unchanged throughout the training. Suppose that each number in embedding matrix and public parameters requires s_1 bits, and that in quantized gradients requires s_2 bits, respectively.

Downloading public parameters requires to transfer $\mathcal{O}(\alpha pKD(D+N_u)Ts_1)$ bits. It takes $\mathcal{O}(\alpha pqDKs_1)$ bits for each party to communicate neighborhood aggregation matrix per iteration, which adds up to $\mathcal{O}(\alpha^2 p^2 qDKTs_1)$ in total.

For gradient transmission, each party is expected to have $\mathcal{O}(|\xi|_1/r)$ nonzero entries in their upload matrix, where $|\xi|_1$ denotes the l_1 norm of public parameters. It takes $\mathcal{O}(\alpha p|\xi|_1 Ts_2/r)$ bits to upload quantized gradients from clients to the server,

Summing up the above processes, the algorithm involves $\mathcal{O}(\alpha pT(KD(D+N_u)s_1 + \alpha pqDKs_1 + |\xi|_1 s_2/r))$ bits of communication cost.

6. Experiment

6.1. Dataset and Experiment Settings

Dataset: We use two benchmark datasets for recommendation, MovieLens-1M[2] (ML-1M) and BookCrossing[3]. For BookCrossing we randomly select 6000 users and 3000 items. The items are divided into non-overlapping groups to simulate the vertical federated setting.

Implementation and Hyper-parameter Setting: Appendix H details the implementation and hyperparameters.

6.2. Experiment Result

6.2.1. Comparison with Different Methods

We compare our proposed method with several centralized and federated recommender system, including: matrix factorization (MF) (Koren et al., 2009), central implementation of GNN (CentralGNN), federated GNN with graph expansion (FedPerGNN) (Wu et al., 2022a), adaption of FedSage and FedSage+ to GNN-based recommender system (Zhang et al., 2021b). We implement the GNN-related methods using the three GNN frameworks introduced in section 3.2.

We compare the methods along four dimensions in table 1: (i) high-order interaction: modeling of high-order connectivity with graph propagation; (ii) gradient protection:

[2]https://grouplens.org/datasets/movielens/1m/

[3]http://www2.informatik.uni-freiburg.de/ cziegler/BX/

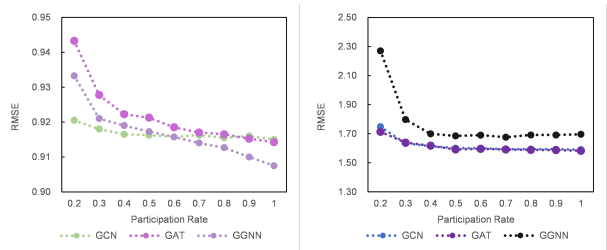
sending perturbed or encrypted gradients instead of raw gradients; (iii) cross graph neighborhood: usage of missing links across parties or subgraphs.

Table 2 summarizes the performance in terms of RMSE for different methods. For VerFedGNN, we use privacy budget $\frac{1}{r} = \frac{1}{3}$, reduced dimension $q = N_u/5$, and participation rate $\alpha = 1$. It can be observed that our proposed method achieves lower RMSE than other federated GNN algorithms in most scenarios, and clearly outperform MF by an average of 4.7% and 18.7% respectively for ML-1M an BookCrossing dataset. The RMSE in VerFedGNN slightly increases over the central implementation, with average percentage difference $\leq 1.8\%$.

6.2.2. Hyper-parameter Studies

We use GCN as an example to study the impact of hyper-parameters on the performace of VerFedGNN.

Participation rate: The participation rate is changed from 0.2 to 1, with results presented in figure 2. Using GCN model, the percentage differences over the fully participation case are within 0.15% for ML-1M and 0.7% for BookCrossing when α reaches to 0.5. The other two models gives RMSE ≤ 0.92 for ML-1M and ≤ 1.2 for BookCrossing when $\alpha > 0.5$.

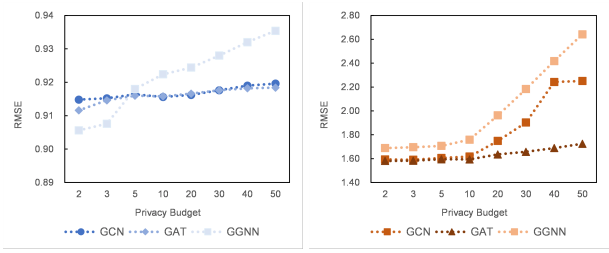


(a) ML-1M (b) BookCrossing

Figure 2. RMSE with varying participation rate α.

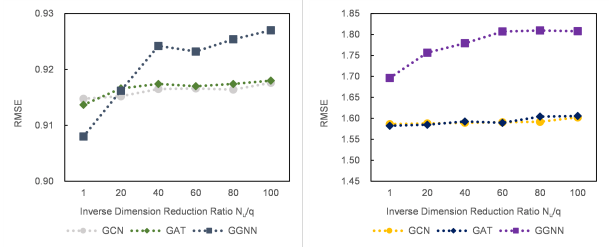
Privacy budget: A smaller the privacy budget $\frac{1}{r}$ suggests that the transmitted gradients leak less user information. Figure 3 presents the tradeoff between privacy budget $\frac{1}{r}$ and model performance. GGNN model is most sensitive to the change in privacy budget, while GAT model remains effective against the increase in $\frac{1}{r}$.

Dimension Reduction: We further analyze the effectiveness of our model with varying dimension reduction ratio q/N_u. As is shown in figure 4, GCN and GAT are more robust to the change in neighborhood dimension q, with error increase by 0.5% for ML-1M and 1.5% for BookCrossing when N_u/q increases to 100.



(a) ML-1M (b) BookCrossing

Figure 3. RMSE with varying privacy budget $\frac{1}{r}$.



(a) ML-1M (b) BookCrossing

Figure 4. RMSE by inverse dimension reduction ratio N_u/q.

6.2.3. De-anonymization Attack

To verify the effectiveness of our model against de-anonymization attack, we simulate this attack to compare the inference accuracy of VerFedGNN with the other two methods using cross graph neighborhood: FedPerGNN and FedSage+. For FedSage+, we match the generated embeddings of honest and adversarial users using equation 11. The attack for VerFedGNN utilized the recovered aggregation embeddings. Specifically, we find the subset of adversarial item embeddings leading to smallest l_1 distance with each users' neighborhood aggregation. Refer to appendix I for more illustrations.

Table 3 reports the attack accuracy for the three federated algorithm using GCN model. The experiment is conducted under three cases regarding the proportion of items rated by the adversarial users p_{ad}. One important observation is that our algorithm greatly reduces the attack accuracy compared with the two baseline methods. FedPerGNN results in highest F1 and precision of 100% as the attacker could match the embeddings exactly with adversarial users.

6.2.4. Communication Cost

Figure 5 presents the communication cost measured in the size of bits to be transferred in each iteration. We find that the communication cost is nearly proportional to user size

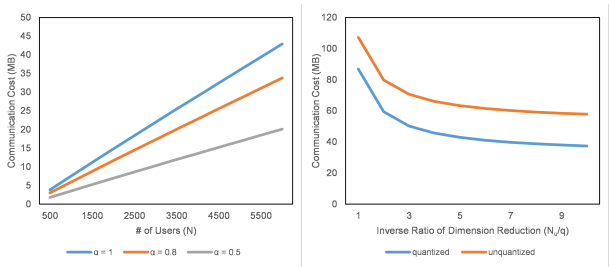
Table 1. Comparison of different approaches

	MF	CentralGNN	FedPerGNN	FedSage	FedSage+	VerFedGNN
High-order interaction	×	√	√	√	√	√
Gradient protection	×	×	√	×	×	√
Cross graph neighborhood	×	×	√	×	√	√
Data storage	Central	Central	Local	Local	Local	Local

Table 2. Performance of different methods. The values denote the $mean \pm standard\ deviation$ of the performance.

	Model	ML-1M	BookCrossing
MF	MF	0.9578 ±0.0016	1.9972 ±0.0063
CentralGNN	GCN	0.9108 ±0.0007	1.5820 ±0.0050
	GAT	0.9062 ±0.0029	1.5478 ±0.0071
	GGNN	0.9046 ±0.0045	1.6562 ±0.0040
FedPerGNN	GCN	0.9282 ±0.0012	1.6892 ±0.0068
	GAT	0.9282 ±0.0017	1.6256 ±0.0048
	GGNN	0.9236 ±0.0023	1.6962 ±0.0050
FedSage	GCN	0.9268 ±0.0012	1.6916 ±0.0118
	GAT	0.9242 ±0.0041	1.6256 ±0.0048
	GGNN	0.9268 ±0.0008	2.6596 ±0.0133
FedSage+	GCN	0.9194 ±0.0041	1.6335 ±0.0065
	GAT	0.9146 ±0.0033	1.6078 ±0.0039
	GGNN	0.9180 ±0.0002	1.8788 ±0.0401
VerFedGNN	GCN	**0.9152 ±0.0013**	**1.5906 ±0.0030**
	GAT	**0.9146 ±0.0010**	**1.5830 ±0.0131**
	GGNN	**0.9076 ±0.0024**	**1.6962 ±0.0050**

N_u and participation rate α. Besides, random projecting the neighborhood aggregation matrix with $q = \frac{1}{5}N_u$ saves the communication bits by 50.6% with gradient quantization, and applying the quantization scheme reduces the communication cost by over 30% when $N_u/q \geq 4$.



(a) User size (b) Dimension

Figure 5. Communication cost by user size and dimension for GCN

6.2.5. Other Studies

For other studies, we simulate the de-anonymization attack against VerFedGNN under the case with and without dimension reduction, and evaluate the model performance when Laplace noise is employed in place of ternary quantization scheme (see Appendix J).

Table 3. Attack accuracy for three federated algorithms using GCN model on ML-1M.

p_{ad}	Methods	Precision	Recall	F1
0.2	FedPerGNN	1.00 ±0.00	0.21 ±0.01	0.34 ±0.01
	FedSage+	0.14 ±0.00	0.03 ±0.01	0.05 ±0.01
	VerFedGNN	0.01 ±0.00	0.01 ±0.00	0.01 ±0.00
0.5	FedPerGNN	1.00 ±0.00	0.49±0.03	0.66 ±0.02
	FedSage+	0.22 ±0.03	0.08 ±0.00	0.11 ±0.00
	VerFedGNN	0.02 ±0.01	0.01 ±0.00	0.01 ±0.00
0.8	FedPerGNN	1.00 ±0.00	0.81±0.01	0.90 ±0.01
	FedSage+	0.26 ±0.02	0.10±0.01	0.14 ±0.01
	VerFedGNN	0.02 ±0.00	0.01 ±0.00	0.02 ±0.00

7. Conclusion

This paper proposes VerFedGNN, a framework for GNN-based recommender systems in a vertical federated setting. The cross-graph interactions are transferred in form of neighborhood aggregation matrix perturbed by random projection. We adopt ternary quantization scheme to protect the privacy of public gradietns. Our approach could learn the relation information across different graphs while preserving users' interaction data. Empirical studies on two benchmark datasets show that: (1) VerFedGNN achieves comparative prediction performance with SOTA privacy preserving GNN models. (2) The neighborhood aggregation combined with random projection significantly reduces the attack accuracy compared with existing cross-graph propagation methods. (3) Optimizing dimension reduction ratio N_u/q and participation rate α could lower the communication cost while maintaining accuracy.

This work opens up new possibilities for the federated GCN-based recommendation. Firstly, it's interesting to develop a scalable federated framework with up to millions of users. Secondly, the framework could be extended to other federated scenarios, such as transfer federated recommender systems with few overlapping nodes (Yang et al., 2020).

References

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

Chen, F., Li, P., Miyazaki, T., and Wu, C. Fedgraph: Federated graph learning with intelligent sampling. *IEEE Transactions on Parallel and Distributed Systems*, 33(8): 1775–1786, 2021.

Du, W., Han, Y. S., and Chen, S. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 2004 SIAM international conference on data mining*, pp. 222–233. SIAM, 2004.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 429–438. IEEE, 2013.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Emiris, I. Z., Karasoulou, A., and Tzovas, C. Approximating multidimensional subset sum and minkowski decomposition of polygons. *Mathematics in Computer Science*, 11 (1):35–48, 2017.

Gao, C., Wang, X., He, X., and Li, Y. Graph neural networks for recommender system. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1623–1625, 2022.

Ghojogh, B., Ghodsi, A., Karray, F., and Crowley, M. Johnson-lindenstrauss lemma, linear and nonlinear random projections, random fourier features, and random kitchen sinks: Tutorial and survey. *arXiv preprint arXiv:2108.04172*, 2021.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. 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*, pp. 639–648, 2020.

Jin, B., Gao, C., He, X., Jin, D., and Li, Y. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 659–668, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kolesnikov, V. Multidimensional subset sum problem. 1997.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Lindenstrauss, W. J. J. Extensions of lipschitz maps into a hilbert space. *Contemp. Math*, 26(189-206):2, 1984.

Liu, K., Kargupta, H., and Ryan, J. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on knowledge and Data Engineering*, 18(1):92–106, 2005.

Liu, Z., Yang, L., Fan, Z., Peng, H., and Yu, P. S. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2021.

Peyré, G., Cuturi, M., et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Pinkas, B., Schneider, T., and Zohner, M. Faster private set intersection based on {OT} extension. In *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 797–812, 2014.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, H., Lian, D., and Ge, Y. Binarized collaborative filtering with distilling graph convolutional networks. *arXiv preprint arXiv:1906.01829*, 2019a.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019b.

Wang, Y. and Başar, T. Quantization enabled privacy protection in decentralized stochastic optimization. *IEEE Transactions on Automatic Control*, 2022.

Wu, C., Wu, F., Lyu, L., Qi, T., Huang, Y., and Xie, X. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 13 (1):1–10, 2022a.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022b.

Yang, L., Tan, B., Zheng, V. W., Chen, K., and Yang, Q. Federated recommendation systems. In *Federated Learning*, pp. 225–239. Springer, 2020.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Zhang, C., Zhang, S., James, J., and Yu, S. Fastgnn: A topological information protected federated learning approach for traffic speed forecasting. *IEEE Transactions on Industrial Informatics*, 17(12):8464–8474, 2021a.

Zhang, K., Yang, C., Li, X., Sun, L., and Yiu, S. M. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems*, 34:6671–6682, 2021b.

Zhou, J., Chen, C., Zheng, L., Wu, H., Wu, J., Zheng, X., Wu, B., Liu, Z., and Wang, L. Vertically federated graph neural network for privacy-preserving node classification. *arXiv preprint arXiv:2005.11903*, 2020.

A. Embedding Update with Neighborhood Aggregation

A.1. Embedding Update for GCN

Suppose the active client c receive $E_p(n_u^k)$ from other parties $p \neq c$. The update function for user embedding is:

$$e_u^{k+1} = \sigma(W^k(e_u^k + \sum_p E_p(n_u^k))) \tag{23}$$

The update for item embedding is conducted locally:

$$Agg_k : \ n_v^k = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{E_c(N_u)N_v}} e_u^k, \ \ Update_k : \ e_v^{k+1} = \sigma(W^k(e_v^k + n_v^k)) \tag{24}$$

A.2. Embedding Update for GAT

Suppose the active client c receive $E_p(n_u^k)$ from other parties $p \neq c$. The update function for user embedding is:

$$e_u^{k+1} = \sigma(W^k(E_c(b_{uu}^k)e_u^k + \sum_p E_p(n_u^k))) \tag{25}$$

The update for item embedding is conducted locally:

$$Agg_k : \ n_v^k = \sum_{u \in \mathcal{N}(v)} b_{vu}^k e_u^k, \ \ Update_k : \ e_v^{k+1} = \sigma(W^k(b_{vv} e_v^k + n_v^k)) \tag{26}$$

where b_{vv}^k and b_{vu}^k are computed as:

$$b_{vu}^k = \frac{exp(\tilde{b}_{vu}^k)}{\sum_{u' \in \mathcal{N}(v) \cup v} exp(\tilde{b}_{vu}^k)} \tag{27}$$

where $b_{vu}^k = Att(e_v^k, e_u^k)$ is computed by an attention function.

A.3. Embedding Update for GGNN

Suppose the active client c receive $E_p(n_u^k)$ from other parties $p \neq c$. The update function for user embedding is:

$$e_u^{k+1} = GRU(e_u^k, \frac{1}{\sum_i M_i} \sum_p E_p(n_u^k) \cdot M_p) \tag{28}$$

The update for item embedding is conducted locally:

$$Agg_k : \ n_v^k = \sum_{u \in \mathcal{N}(v)} \frac{1}{N_v} e_u^k, \ \ Update_k : \ e_v^{k+1} = GRU(e_v^k, n_v^k) \tag{29}$$

B. Proof of Lemma 5.2

Proof. The proof follows by a proper Gaussian elimination on the system of linear equations. See, for instance, Theorem 4.3 in (Du et al., 2004). □

C. Proof of Theorem 5.3

Proof. According to Theorem 4.4 in (Liu et al., 2005), when $2q - 1 \leq m$, a submatrix Φ_k formed by removing any $q - 1$ columns from Φ has full rank with probability 1, i.e., the linear system is $(q - 1)$-secure with probability 1. Hence, any nonzero linear combination of the row vectors in Φ contains at least $q - 1$ nonzero elements. According to lemma 5.2, we cannot find $q - 1$ linearly equations that solve these variables. Therefore, the solutions to any variable in x are infinite. □

D. Interpretation of Theorem 5.4

D.1. Explanation of Remark 5.5

The original definition of differential privacy uses l_1 norm to denote the number of different records for two sets. In our study, the inputs are continuous and thus we use l_p-norm to measure the l_p-distance between two vectors. The neighboring datasets can be interpreted as vectors close to each other in terms of l_p-distance.

D.2. Explanation of $(0, \frac{1}{r})$-DP

The privacy budget is controlled by two parameters ϵ and δ. The $(0, 1/r)$-differential privacy is formulated as:

$$P(M(x) \in S) \leq P(M(y) \in S) + \frac{1}{r} \tag{30}$$

Therefore, the $(0, \frac{1}{r})$-DP suggests that the absolute difference of probability density at each point differs by at most $\frac{1}{r}$.

D.3. Lower Bound of Reconstruction Attack Error

We take $p = \infty$ as an example to derive a proof for the lower bound of reconstruction attack error for the l_p norm.

Theorem D.1. *Let $h = M(x)$ be the model output given input vector x, and $\hat{x}(h)$ be the reconstructed input on observing h. For a (ϵ, δ)-DP mechanism M with l_1 replaced by l_∞, the reconstruction error defined as mean square error (MSE) is lower bounded by:*

$$\mathbb{E}[\|\hat{x}(h) - x\|_2^2] \geq O\left(\frac{\sum_i \theta_i^2}{e^{2\epsilon} + e^{\epsilon}\delta^2 - 1}\right) \tag{31}$$

where $\theta_i = \inf_x |\partial\mu(x)_i / \partial x_i|$ and $\mu(x) = \mathbb{E}[\hat{x}(h)]$.

Proof. The MSE is lower bounded by:

$$\mathbb{E}[\|\hat{x}(h) - x\|_2^2] \geq \sum_i Var\left(\hat{x}(h)_i\right) \tag{32}$$

Then we examine the bound of $Var\left(\hat{x}(h)_i\right)$. From Hammersley-Chapman-Robbins Bound, we have:

$$Var\left(\hat{x}(h)_i\right) \geq \frac{(\mu(x+e_i)_i - \mu(x)_i)^2}{\mathbb{E}[(p(h;x+e_i)/p(h;x) - 1)^2]} = \frac{(\mu(x+e_i)_i - \mu(x)_i)^2}{e^{2\epsilon} + e^{\epsilon}\mathbb{E}[2\delta/p(h;x) + \delta^2/p(h;x)^2] - 1} \tag{33}$$

where $\mathbb{E}[\cdot]$ is the expectation taken over $p(h;x)$, $p(h;x)$ is the density function of h given x, and e_i is the standard basis vector with ith coordinate equal to 1.

Therefore, the MSE is lower bounded by:

$$\mathbb{E}[\|\hat{x}(h) - x\|_2^2] \geq O\left(\frac{\sum_i \theta_i^2}{e^{2\epsilon} + e^{\epsilon}\delta^2 - 1}\right) \tag{34}$$

□

Remark D.2. For $\epsilon = 0$, we have that:

$$\mathbb{E}[\|\hat{x}(h) - x\|_2^2] \geq O\left(\sum_i \theta_i^2 / \delta^2\right) \tag{35}$$

E. NP-hardness of Finding Missing Neighbors from Reconstructed Matrix

E.1. Missing Neighbors for GCN

Let $E_p(\hat{n}_u^k)$ be the recovered neighborhood for user u in party p. Denote $\hat{e}_{v_j}^k / \sqrt{E_p(N_u)}$ as the reconstructed embedding from adversarial user j for $v_j \in \mathcal{N}_{adv}$. The attacker should find the subset $S \in \mathcal{N}_{adv}$ such that $\sum_{v_j \in S} \hat{e}_{v_j}^k / \left(\sqrt{E_p(N_u)}\sqrt{|S|}\right) = E_p(\hat{n}_u^k)$. The inference attack for GGNN can be summarized as a multi-dimensional subset squared-root average problem.

Problem E.1 (k-dimensional Subset Squared-root Average (kD-SSA)). Input: a set of vectors $S = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$, for $k \geq 1$, and a target vector $t \in \mathbb{Z}^k$. Output: YES if there exists a subset $S' \subseteq S$ such that $\sum_{i \in S'} n^i / \sqrt{|S'|} = t$, and NO otherwise.

We claim the NP-hardness of the problem.

Theorem E.1. *The kD-SSA problem is NP-complete.*

Proof. Prior literature showed that the k-dimensional Subset Sum (kD-SS) problem is NP-complete (Emiris et al., 2017; Kolesnikov, 1997).

Problem E.2 (k-dimensional Subset Sum (kD-SS)). Input: a set of vectors $S = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$, for $k \geq 1$, and a target vector $t \in \mathbb{Z}^k$. Output: YES if there exists a subset $S' \subseteq S$ such that $\sum_{i \in S'} n^i = t$, and NO otherwise.

We start with the reduction of kD-SS to Size M kD-SS.

Problem E.3 (Size M k-dimensional Subset Sum (M-kD-SS)). Input: a set of vectors $S = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$, for $k \geq 1$, a target subset size M and a target vector $t \in \mathbb{Z}^k$. Output: YES if there exists a subset $S' \subseteq S$ such that $\sum_{i \in S'} n^i = t$ and $|S'| = M$, and NO otherwise.

It's clear that M-kD-SS $\in NP$ as we can verity that a subset equals t and has size M in polynomial time. Next we show the reduction of kD-SS to M-kD-SS.

Let $S_1 = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$ and t be the input to kD-SS. We form S_2 by adding n zero-vectors in to S_1. Let S_2, $M = n$, and t be the input to M-kD-SS. Let S_1' be the solution to kD-SS, and S_2' is constructed by adding $n - |S_1'|$ to S_1'. The reduction works clearly in polynomial time.

Next, we claim that $S_1' \in$ kD-SS iif $S_2' \in$ M-kD-SS, i.e., S_1' is a solution to kD-SS iif S_2' is a solution to M-kD-SS.

$\Rightarrow$: If the elements in S_1' sums up to t, the same is true for S_2' that's of size n. Therefore, S_2' is a solution to M-kD-SS.

$\Leftarrow$: If S_2' is a solution to M-kD-SS, then the solution to kD-SS S_1' can be formed by removing the zero vectors in S_2'.

Now, we have demonstrated the NP-completeness of M-kD-SS, and will return back to the kD-SSA problem.

kD-SSA $\in NP$ as we can verity that a subset sum divided by its square-root size equals to t in polynomial time. Next we show the reduction of M-kD-SS to kD-SSA.

Let $S_1 = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$, M and t be the input to M-kD-SS. We form S_2 by adding to S_1 a vector v^M consisting of $MaxN \cdot \sqrt{M+1}$, such that:

$$MaxN \gg \frac{|t_j| + \sum_{i \in S_1} |n_j^i|}{|\sqrt{r_1} - \sqrt{r_2}|}, \forall r_1 \neq r_2, r_1, r_2 \in [1, |S_2|], \forall j \in [1, k] \tag{36}$$

$MaxN$ is much larger than any subset sum of absolute values in $S_1 \cup \{t\}$. Let S_2, and $t/\sqrt{M+1} + MaxN$ be the input to kD-SA. Let S_1' be the solution to M-kD-SS, and S_2' is constructed by adding vector v^M to S_1'. The reduction works clearly in polynomial time.

We proceed to claim that $S_1' \in$ M-kD-SS iif $S_2' \in$ kD-SSA, i.e., S_1' is a solution to M-kD-SS iif S_2' is a solution to kD-SSA.

$\Rightarrow$: If S_1' is a solution to M-kD-SS, then the square-root average of S_2' is given by:

$$\left(\sum_{i \in S_1'} n_j^i + v_j^M \right) \cdot \frac{1}{\sqrt{M+1}} = \frac{t_j}{\sqrt{M+1}} + MaxN \tag{37}$$

for $1 \leq j \leq k$. Thus S_2' is a solution to kD-SA.

$\Leftarrow$: Let S_2' be a solution to kD-SA. Suppose $v^M \notin S_2'$, then:

$$\left(\sum_{i \in S_2'} n_j^i \right) \cdot \frac{1}{\sqrt{|S_2'|}} = \frac{t_j}{\sqrt{M+1}} + MaxN \tag{38}$$

for $1 \leq j \leq k$. The equation doesn't hold since $\frac{t_j}{M+1} + MaxN$ should be much larger than the sum of any subsets of $S_2 \backslash v^M$. The argument shows that $v^M \in S_2'$, giving the following expression:

$$\left(MaxN \cdot \sqrt{M+1} + \sum_{i \in S_2' \backslash v^M} n_j^i \right) \cdot \frac{1}{\sqrt{|S_2'|}} = \frac{t_j}{\sqrt{M+1}} + MaxN$$
$$\Updownarrow \tag{39}$$
$$MaxN \left(\sqrt{M+1} - \sqrt{|S_2'|} \right) = \frac{t_j |S_2'|}{\sqrt{M+1}} - \sum_{i \in S_2' \backslash v^M} n_j^i$$

for $1 \leq j \leq k$. Suppose that $|S_2'|$ is not of size $M+1$, then the equation doesn't hold given that $MaxN$ should be a very large number. Therefore, we prove by contradiction that S_2' is of size $M+1$ and includes v^M. Then the S_1' formed by removing v^M should be a solution to M-kD-SS. □

E.2. Missing Neighbors for GAT

Let $E_p(\hat{n}_u^k)$ be the recovered neighborhood for user u in party p. Denote $b_{uv}^k \hat{e}_{v_j}^k$ as the reconstructed embedding from adversarial user j for $v_j \in \mathcal{N}_{adv}$. One challenge to the inference attack is the unknown coefficient b_{uv}^k. We consider a simpler problem where b_{uv}^k is know in advance for every pair of nodes, and show that even the simplified version belongs to NP-complete.

Given the coefficient b_{uv}^k, the attacker can compute $\hat{e}_{v_j}^k$ for $v_j \in \mathcal{N}_{adv}$. Then it should find the subset $S \in \mathcal{N}_{adv}$ such that $\sum_{v_j \in S} b_{uv_j}^k \hat{e}_{v_j}^k = E_p(\hat{n}_u^k)$. This is essentially the k-dimensional Subset Sum (kD-SS) problem that belongs to NP-complete.

E.3. Missing Neighbors for GGNN

Let $E_p(\hat{n}_u^k)$ be the recovered neighborhood for user u in party p. Denote $\hat{e}_{v_j}^k$ as the reconstructed embedding from adversarial user j for $v_j \in \mathcal{N}_{adv}$. The attacker should find the subset $S \in \mathcal{N}_{adv}$ such that $\sum_{v_j \in S} \hat{e}_{v_j}^k / |S| = E_p(\hat{n}_u^k)$. The inference attack for GGNN can be summarized as the multi-dimensional subset average problem.

Problem E.4 (k-dimensional Subset Average (kD-SA)). Input: a set of vectors $S = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$, for $k \geq 1$, and a target vector $t \in \mathbb{Z}^k$. Output: YES if there exists a subset $S' \subseteq S$ such that $\sum_{i \in S'} n^i / |S'| = t$, and NO otherwise.

We aim to show the NP-hardness of the problem.

Theorem E.2. *The kD-SA problem is NP-complete.*

Proof. As we have demonstrated the NP-completeness of M-kD-SS in the proof of Theorem E.1, we will show its reduction to the kD-SA problem.

kD-SA $\in NP$ as we can verity that a subset averages to t in polynomial time. Next we show the reduction of M-kD-SS to kD-SA.

Let $S_1 = \{n^i | 1 \leq i \leq n\} \subset \mathbb{Z}^k$, M and t be the input to M-kD-SS. We form S_2 by adding to S_1 a vector v^M consisting of $MaxN \cdot (M+1)$, such that $MaxN$ is much larger than any subset sum of absolute values in $S_1 \cup \{t\}$. Let S_2, and $t/(M+1) + MaxN$ be the input to kD-SA. Let S_1' be the solution to M-kD-SS, and S_2' is constructed by adding vector v^M to S_1'. The reduction works clearly in polynomial time.

We proceed to claim that $S_1' \in$ M-kD-SS iif $S_2' \in$ kD-SA, i.e., S_1' is a solution to M-kD-SS iif S_2' is a solution to kD-SA.

$\Rightarrow$: If S_1' is a solution to M-kD-SS, then the average of S_2' is given by:

$$\left(\sum_{i \in S_1'} n_j^i + v_j^M \right) \cdot \frac{1}{M+1} = \frac{t_j}{M+1} + MaxN \tag{40}$$

for $1 \leq j \leq k$. Thus S_2' is a solution to kD-SA.

$\Leftarrow$: Let S_2' be a solution to kD-SA. Suppose $v^M \notin S_2'$, then:

$$\left(\sum_{i\in S_2'} n_j^i\right)\cdot\frac{1}{|S_2'|}=\frac{t_j}{M+1}+MaxN \tag{41}$$

for $1\le j\le k$. The equation doesn't hold since $\frac{t_j}{M+1}+MaxN$ should be much larger than any other elements in S_2. The argument shows that $v^M\in S_2'$, giving the following expression:

$$\begin{gathered}\left(MaxN\cdot(M+1)+\sum_{i\in S_2'\setminus v^M} n_j^i\right)\cdot\frac{1}{|S_2'|}=\frac{t_j}{M+1}+MaxN\\ \Updownarrow\\ MaxN\left(M+1-|S_2'|\right)=\frac{t_j|S_2'|}{M+1}-\sum_{i\in S_2'\setminus v^M} n_j^i\end{gathered} \tag{42}$$

for $1\le j\le k$. Suppose that $|S_2'|$ is not of size $M+1$, then the equation doesn't hold since $MaxN\gg|\frac{t_j|S_2'|}{M+1}-\sum_{i\in S_2'\setminus v^M} n_j^i|$. Therefore, we prove by contradiction that S_2' is of size $M+1$ and includes v^M. Then the S_1' formed by removing v^M should be a solution to M-kD-SS. □

F. Proof of Theorem 5.6

Proof. The reconstruction MSE can be written as:

$$\begin{gathered}\mathbb{E}_\Phi[\|\Phi^T\Phi X-X\|_F^2]=\mathbb{E}_\Phi[\sum_i\|\Phi^T\Phi X_i-X_i\|_F^2]=\\ \sum_i(X_i^T\mathbb{E}_\Phi[\Phi^T\Phi\Phi^T\Phi]X_i-2X_i^TE[\Phi^T\Phi]X_i+X_i^TX_i)\end{gathered} \tag{43}$$

where X_i denote the ith column of X.

Then we can compute the expectation of the random matrix. Let $A=\Phi^T\Phi\Phi^T\Phi$, and $B=\Phi^T\Phi$. The expectation of elements in A is:

$$\mathbb{E}(A_{ij})=\begin{cases}\frac{m+1}{p}+1, & i=j\\ 0, & i\neq j\end{cases} \tag{44}$$

The expectation of elements in B is:

$$\mathbb{E}(B_{ij})=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \tag{45}$$

Plug in the expectation of A and B, the MSE is computed as:

$$\mathbb{E}_\Phi[\|\Phi^T\Phi X-X\|_F^2]=\frac{(m+1)}{p}\|X\|_F^2 \tag{46}$$

□

G. Convergence Analysis of Ternary Quantization Mechanism

This section provides the convergence analysis of the ternary quantization scheme. The loss function in 5 can be decomposed as:

$$\mathcal{L}=f(x)=\sum_p\underbrace{\left(\sum_{(u,v)\in\mathcal{U}\times\mathcal{V}_p}(\hat{r}_{uv}-r_{uv})^2+\frac{1}{M}\sum_{v\in\mathcal{V}_p}\|e_v^0\|^2\right)}_{f_p(x)}+\frac{1}{N}\sum_{u\in\mathcal{U}}\|e_u^0\|^2=\sum_p f_p(x)+\frac{1}{N}\sum_{u\in\mathcal{U}}\|e_u^0\|^2 \tag{47}$$

The second term can be ignored since it can be computed at the server. Denote g_p^t as the unbiased estimation of gradients using the raw aggregation matrix, $\tilde{g}_p^t$ as the biased estimation of gradients using the projected matrix $\hat{X}^k$, $k \in [0, K-1]$, and $q(\tilde{g}_p^t)$ as the gradients perturbed by ternary quantization. We start with some properties of the ternary scheme from Definition 4.2.

Lemma G.1. *Under the ternary quantization scheme given by Definition 4.2, it holds that:*

$$\mathbb{E}[q(\tilde{g}_p^t)] = \tilde{g}_p^t,\ \mathbb{E}[q(\tilde{g}_p^t) - \tilde{g}_p^t] \leq |\zeta_p| r^2,\ \ \forall p, t \tag{48}$$

where $|\zeta_p|$ denotes the number of parameters.

Proof. The proof follows by directly computing $\mathbb{E}[q_i]$ and $\mathbb{E}[q(\tilde{g}_p^t) - \tilde{g}_p^t]$:

$$\begin{aligned} \mathbb{E}[q(\tilde{g}_p^t)] &= \text{sign}(\tilde{g}_p^t) \cdot r \cdot \frac{|\tilde{g}_p^t|}{r} = \tilde{g}_p^t \\ \mathbb{E}[q(\tilde{g}_p^t) - \tilde{g}_p^t] &= \sum_i r|x_i| \leq \sum_i r^2 = |\zeta_p| r^2 \end{aligned} \tag{49}$$

□

We further need the following assumptions for the proof.

Assumption G.2. (1) Each party p has Lipschitz continuous function $f_p(\cdot)$ with Lipschitz gradients

$$\|\nabla f_p(x) - \nabla f_p(y)\| \leq L_1 \|x - y\| \tag{50}$$

where x and y denote any vectors of public parameters, and (1) always has at least one optimal solution $x*$, i.e., $\sum_{i=1}^{p} \nabla f_p(x*) = 0$.

(2) Denote the gradient with regards to parameter as a function of all parameters and the received neighborhood matrix:

$$g_p^t = h_{\zeta_p^t}(\xi, X^0, ..., X^{K-1}), \tilde{g}_p^t = h_{\zeta_p^t}(\xi, \hat{X}^0, ..., \hat{X}^{K-1}), \forall t, p \tag{51}$$

We assume that $h_{\zeta_p^t}(x)$ is a Lipschitz function:

$$|h_{\zeta_p^t}(x) - h_{\zeta_p^t}(y)| \leq L_2 \|x - y\|, \forall p, t \tag{52}$$

(3) The Frobenius norm of aggregated neighborhood matrix is bounded by G:

$$\|X^k\|_F^2 \leq G, \forall k \in [0, K-1] \tag{53}$$

(4) There exists a constant M such that the bias of gradient estimation is bounded by:

$$\|b_p^t\|^2 + \sigma^2 \leq M \|\nabla f_p(x^t)\|^2, \forall p, t \tag{54}$$

where $b_p^t = \mathbb{E}[\tilde{g}_p^t - g_p^t]$ is the bias of gradient estimation.

Based on the above assumptions, we can have the following convergence guarantee.

Theorem G.3. *Under Assumption G.2, if each party sends to server the unbiased quantized gradients given by Definition 4.2 for aggregation and update, we get:*

$$\min_t \mathbb{E}[\|\nabla f(x^t)\|^2] \leq O\left(\frac{|\zeta| r^2 + K(m+1) L_2 G / p}{\log T}\right) \tag{55}$$

by choosing the learning rate $\gamma^t = \frac{1}{(t+1) M L_1}$.

Proof. Based on the Lipschitz smoothness assumption, it holds that:

$$f(x^{t+1}) \leq f(x^t) - \gamma^t \langle \nabla f(x^t), \sum_p q(\tilde{g}_p^t) \rangle + \frac{L_1 (\gamma^t)^2 \| \sum_p q(\tilde{g}_p^t) \|^2}{2} \tag{56}$$

By taking the expectation, we have:

$$\begin{aligned} \mathbb{E}[f(x^{t+1})] \leq \mathbb{E}[f(x^t)] - \gamma^t \mathbb{E}[\|\nabla f(x^t)\|^2] - \gamma^t \sum_p \mathbb{E}\langle f(x^t), b_p^t \rangle \\ + \frac{L_1(\gamma^t)^2}{2} \left(\sum_p \left[\mathbb{E}[\|q(\tilde{g}_p^t) - \tilde{g}_p^t\|^2] + \mathbb{E}[\|f(x^t) + b_p^t\|^2] \right] \right) \end{aligned} \tag{57}$$

By selecting $\gamma^t = \frac{1}{(t+1)ML_1}$, it follows that:

$$\left(\frac{1}{(t+1)ML_1} - \frac{1}{2(t+1)^2 M^2 L_1} \right) \mathbb{E}[\|\nabla f(x^t)\|^2] \leq \mathbb{E}[f(x^t)] - \mathbb{E}[f(x^{t+1})] + \frac{\sum_p \left(\mathbb{E}[\|q(\tilde{g}_p^t) - \tilde{g}_p^t\|^2] + \mathbb{E}[\|b_p^t\|^2] \right)}{2(t+1)^2 M^2 L_1} \tag{58}$$

Aggregating both sides over all iterations, we have:

$$\min_t \mathbb{E}[\|\nabla f(x^t)\|^2] \leq O\left(\frac{\sum_p \left(\mathbb{E}[\|q(\tilde{g}_p^t) - \tilde{g}_p^t\|^2] + \mathbb{E}[\|b_p^t\|^2] \right)}{\log T} \right) \tag{59}$$

Next, we examine the bound of $\mathbb{E}[\|b_p^t\|^2]$. Under the Lipschitz assumption of $h_{\zeta_p^t}(x)$, it holds that:

$$\mathbb{E}[\|b_p^t\|^2] \leq \frac{K(m+1)L_2 G}{p} \tag{60}$$

Plug equation 48 and 60 into the convergence function, we have:

$$\min_t \mathbb{E}[\|\nabla f(x^t)\|^2] \leq O\left(\frac{|\zeta| r^2 + K(m+1)L_2 G/p}{\log T} \right) \tag{61}$$

□

H. Implementation and Hyper-parameters Setting

The experiment is implemented on Ubuntu Linux 20.04 server with 16-core CPU and 64GB RAM, where the programming language is Python.

Cross-validation is adopted to tune the hyper-parameter, where the training-validation-testing ratio is 60%-20%-20%. Each experiment is run for five rounds. The model parameters are updated using Adagrad algorithm (Duchi et al., 2011). Based on the hyper-parameter optimization, we set embedding dimension D to 6, layer size K to 2, learning rate η to 0.05, and neighbor threshold thd to 4 for ML-1M and 8 for BookCrossing. We use sigmoid as the activation function.

We consider privacy parameter r from 2 to 50, inverse dimension reduction ratio N_u/q from 1 to 100, and participation rate from 0.2 to 1. The immediate gradients are clipped within $[-0.5, 0.5]$ so that $\|g_1 - g_2\|_\infty \leq 1$ before applying ternary quantization.

I. De-anonymization Attack for VerFedGNN

I.1. Attack for GCN

The attacker can obtain the v_j's reconstructed weighted embedding $\hat{e}_{v_j}^0 = \text{Rec}(e_{v_j}^0/\sqrt{N_{v_j}})$ from adversarial user j for $v_j \in \mathcal{N}_{adv}$. For any honest user u, given their perturbed embedding aggregation for the initial layer $E_p(\hat{n}_u^0)$, the attacker

could develop a mixed integer programming problem, with binary variables $x^p \in \{0,1\}^{N_v^p}$ denoting the presence of item v in user u's neighborhood. Denote $\mathcal{N}_{adv}^p$ as the items rated by the adversarial users in party p.

$$\begin{aligned} \text{objective: } \|E_p(\hat{n}_u^0) - \sum_{v_j \in \mathcal{N}_{adv}^p} x_{v_j}^p \text{Rec}(e_{v_j}^0 / \sqrt{N_{v_j}}) / \sqrt{c})\|_1 \\ \text{s.t.: } \sum_{v_j \in \mathcal{N}_{adv}} x_{v_j} = c \end{aligned} \tag{62}$$

for $c \in [1, N_v^p]$. As enumerating c from 1 to N_v^p has complexity of $\mathcal{O}(2^{N_v^p})$, we set the upper limit of c as 3 to make it computationally feasible.

I.2. Attack for GAT

The attacker can obtain the v_j's reconstructed weighted embedding $\hat{e}_{v_j}^0 = \text{Rec}(e_{v_j}^0 E_p(b_{uv}^0))$ from adversarial user j for $v_j \in \mathcal{N}_{adv}$. One challenge is that the attacker couldn't obtain the $E_p(b_{uv}^0),\ u \in \mathcal{N}(u),\ v \in \mathcal{N}^p(v)$ to find the matched subset items. A efficient solution is to estimate $E_p(b_{uv}^0)$ with attacker's local average of connected tuples, and obtain the estimated $\hat{e}_{v_j}^0$. The attacker could develop a integer programming.

$$\text{objective: } \|E_p(\hat{n}_u^0) - \sum_{v_j \in \mathcal{N}_{adv}^p} x_{v_j}^p \bar{b}_{uv} \hat{e}_{v_j}^0\|_1 \tag{63}$$

where $\bar{b}_{uv}$ denote the average of b_{uv} from party p's view.

I.3. Attack for GNN

The attacker can obtain the v_j's reconstructed weighted embedding $\hat{e}_{v_j}^0 = \text{Rec}(e_{v_j}^0)$ from adversarial user j for $v_j \in \mathcal{N}_{adv}$. Given honest user u's aggregated embedding $E_p(\hat{n}_u^0)$, the attacking party could develop a integer programming similar to equation 62.

$$\begin{aligned} \text{objective: } \|E_p(\hat{n}_u^0) - \sum_{v_j \in \mathcal{N}_{adv}^p} x_{v_j}^p \text{Rec}(e_{v_j}^0)/c)\|_1 \\ \text{s.t.: } \sum_{v_j \in \mathcal{N}_{adv}} x_{v_j} = c \end{aligned} \tag{64}$$

for $c \in [1, N_v^p]$. We set the upper limit of c as 3 to make it computationally feasible.

J. Experiment Results of Other Studies

J.1. Impact of Random Projection on De-anonymization attack

To investigate the effectiveness of dimension reduction mechanism in terms of privacy protection, we simulate the de-anonymization attack against VerFedGNN under the case with and without dimension reduction. The impact is conducted using GCN and GGNN models as in GAT the unknown b_{uv} hinders the launch of our attack. Without random projection, the attach challenge comes from two sources: (i) it's computationally infeasible to enumerate the combination of all items, and thus we limit the rated item size no larger than 3 from each party; (ii) only a proportion of item is accessed by the adversarial users.

The results are reported in Table 4 and 5 assuming $p_{ad} = 0.5$. It can be observed that for BookCrossing, F1 is significantly reduced by more than 70% for both models. For ML-1M, F1 is reduced by 12% and 30% for GCN and GGNN models, respectively. The higher accuracy of BookCrossing might result from its sparser interaction matrix that makes it more likely to infer the rated items.

Table 4. Attack accuracy with and without random projection on ML-1M dataset

	Models	Precision	Recall	F1
With Random Projection	GCN	0.0176 ± 0.0074	0.0107 ± 0.0056	0.0132 ± 0.0064
	GGNN	0.0210 ± 0.0041	0.0134 ± 0.0035	0.0160 ± 0.0027
Without Random Projection	GCN	0.0395 ± 0.0102	0.0097 ± 0.0038	0.0150 ± 0.0051
	GGNN	0.0511 ± 0.0125	0.0148 ± 0.0050	0.0228 ± 0.0069

Table 5. Attack accuracy with and without random projection on BookCrossing dataset

	Models	Precision	Recall	F1
With Random Projection	GCN	0.0026 ± 0.0010	0.0071 ± 0.0052	0.0037 ± 0.0018
	GGNN	0.0033 ± 0.0002	0.0064 ± 0.0006	0.0044 ± 0.0002
Without Random Projection	GCN	0.0361 ± 0.0139	0.0077 ± 0.0027	0.0127 ± 0.0045
	GGNN	0.0500 ± 0.0176	0.0224 ± 0.0108	0.0305 ± 0.0129

J.2. Adding Laplace Noise on Gradients

we evaluate the model performance of our framework when Laplace noise is added to the gradients in place of ternary quantization scheme. We set $\epsilon = 1$ for Laplace mechanism and $r = \frac{1}{3}$ for ternary quantization scheme. The results are presented in the Table 6.

Table 6. Performance of different methods. The values denote the $mean \pm standard\ deviation$ of the performance.

	Model	ML-1M	BookCrossing
Laplace	GCN	0.9594 ± 0.0014	1.7360 ± 0.0114
	GAT	0.9318 ± 0.0013	1.6936 ± 0.0191
	GGNN	0.9304 ± 0.0008	1.7042 ± 0.0164
Ternary Quantization	GCN	0.9152 ± 0.0013	1.5906 ± 0.0030
	GAT	0.9146 ± 0.0010	1.5830 ± 0.0131
	GGNN	0.9076 ± 0.0024	1.6962 ± 0.0050

J.3. Scalability of VerFedGNN

To validate the scalability of VerFedGNN, we conducted supplementary experiments on Yelp dataset[4], with 6,990,280 ratings, 1,987,929 users, and 150,346 items. We followed most of the hyper-parameters in Appendix H, except that embedding dimension $D = 20$. The model is successfully deployed on the dataset. We use GCN model as an example to explain the performance.

VerFedGNN achieves RMSE of 1.312, a small drop from 1.302 of CentralGNN while still a significant improve from 1.41 of MF method. In terms of computation cost, the per epoch computation time is around 18 seconds on client side and 5 seconds on server side.

The per iteration communication cost is 660.9MB per party at participation of 0.5, adding up to 6609MB for all parties. The transmission time required for this communication cost is 89 seconds under a per-client bandwidth of 100MB/s. The communication time could be reduced by having each party selecting a portion of common users in each round, rather than updating on on all users. For example, by choosing $1/20$ users in each iteration, the communication time would be reduced to 4.5 second. The method for conducting user selection will be the subject of future research.

[4]https://www.yelp.com/dataset

Table 7. Attack accuracy against gradients using GCN model on ML-1M.

p_{ad}	Methods	Precision	Recall	F1
0.2	FedPerGNN	0.0832 ± 0.026	0.0094 ± 0.003	0.0168 ± 0.006
	FedSage+	0.2383 ± 0.102	0.0116 ± 0.003	0.0220 ± 0.006
	VerFedGNN	0.0527 ± 0.007	0.0030 ± 0.0012	0.0057 ± 0.002
0.5	FedPerGNN	0.1033 ± 0.013	0.0105 ± 0.001	0.0191 ± 0.003
	FedSage+	0.2775 ± 0.067	0.0123 ± 0.002	0.0236 ± 0.004
	VerFedGNN	0.0675 ± 0.021	0.0036 ± 0.003	0.006 ± 0.005
0.8	FedPerGNN	0.1061 ± 0.019	0.0018 ± 0.004	0.0210 ± 0.007
	FedSage+	0.1485 ± 0.016	0.0119 ± 0.005	0.0251 ± 0.008
	VerFedGNN	0.0623 ± 0.026	0.0037 ± 0.002	0.0070 ± 0.003

J.4. De-anonymization Attack against Gradients

To examine the effectiveness of gradient protection, we ran experiments on de-anonymization attack against the user embedding gradients, with steps given as followed:

- Obtain the gradients for adversarial users.
- Find a subset of adversarial users such that their gradient sum is closest to the victim's gradient. (For ternary quantization, we normalized the gradient sum to $[-1, 1]$).
- The inferred items are those rated by the adversarial users.

Table 7 compares the attack accuracy under three federated methods using GCN model on ML-1M datasets. The ternary quantization mechanism significantly lowers the risk from inference attack.