

# Rethinking Byzantine Robustness in Federated Recommendation from Sparse Aggregation Perspective

Zhongjian Zhang<sup>1\*</sup>, Mengmei Zhang<sup>2\*</sup>, Xiao Wang<sup>3</sup>, Lingjuan Lyu<sup>4</sup>  
Bo Yan<sup>1</sup>, Junping Du<sup>1</sup>, Chuan Shi<sup>1†</sup>

<sup>1</sup>Beijing University of Posts and Telecommunications, <sup>2</sup>China Telecom Bestpay, <sup>3</sup>Beihang University, <sup>4</sup>Sony AI  
zhangzj@bupt.edu.cn, zhangmengmei@bestpay.com.cn, xiao\_wang@buaa.edu.cn, lingjuan.lv@sony.com  
{boyan, junpingdu, shichuan}@bupt.edu.cn

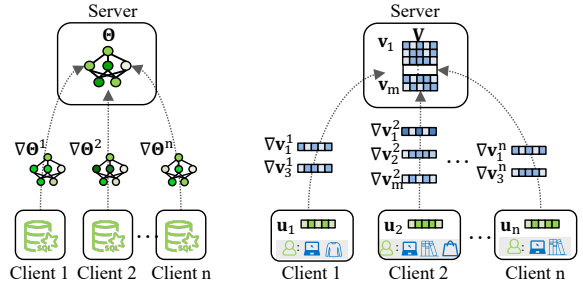
## Abstract

To preserve user privacy in recommender systems, federated recommendation (FR) based on federated learning (FL) emerges, keeping the personal data on the local client and updating a model collaboratively. Unlike FL, FR has a unique sparse aggregation mechanism, where the embedding of each item is updated by only partial clients, instead of full clients in a dense aggregation of general FL. Recently, as an essential principle of FL, model security has received increasing attention, especially for Byzantine attacks, where malicious clients can send arbitrary updates. The problem of exploring the Byzantine robustness of FR is particularly critical since in the domains applying FR, e.g., e-commerce, malicious clients can be injected easily by registering new accounts. However, existing Byzantine works neglect the unique sparse aggregation of FR, making them unsuitable for our problem. Thus, we make the first effort to investigate Byzantine attacks on FR from the perspective of sparse aggregation, which is non-trivial: it is not clear how to define Byzantine robustness under sparse aggregations and design Byzantine attacks under limited knowledge/capability. In this paper, we reformulate the Byzantine robustness under sparse aggregation by defining the aggregation for a single item as the smallest execution unit. Then we propose a family of effective attack strategies, named **Spattack**, which exploit the vulnerability in sparse aggregation and are categorized along the adversary’s knowledge and capability. Extensive experimental results demonstrate that Spattack can effectively prevent convergence and even break down defenses under a few malicious clients, raising alarms for securing FR systems.

## Introduction

As an essential way to alleviate information overload, recommender systems are widely used in e-commerce (Ying et al. 2018), media (Wang et al. 2018; Wu et al. 2019a), and social network (Fan et al. 2019), recommending items that users may be interested in. Despite the remarkable success, conventional recommender systems require centrally storing users’ personal data for training, increasing privacy risks.

Recently, federated learning (FL) (McMahan et al. 2016) has emerged as a privacy-preserving paradigm and success-



(a) Dense aggregation in general FL. (b) Sparse aggregation in FR.

Figure 1: Comparisons between dense aggregation of general FL and the unique sparse aggregation of FR.

fully applied to the recommendation area. In federated recommendation (FR) (Sun et al. 2024; Luo, Xiao, and Song 2022), the global item embeddings are uploaded to a central server for aggregation. Meanwhile, each user’s interaction data and privacy features are kept on the local client. In this way, the privacy of local data is well protected.

Unlike general FL systems, FR has a unique sparse aggregation mechanism. As shown in Fig. 1, for general FL, each element (circle) of model parameters can be updated by all  $n$  clients, named dense aggregation. While for FR, the interactions of users and items are usually sparse (Ma et al. 2008), resulting in each item’s embedding can only be updated by partial clients. For example, client  $n$  can only produce and send substantive gradients  $\{\nabla v_1^n, \nabla v_3^n\}$  for its interacted items  $\{v_1, v_3\}$ . For the remaining items, the updates are zero vectors or empty, named sparse aggregation.

By far, FR has provided satisfactory performance without collecting users’ private data, extending recommendation applications to privacy-sensitive scenarios. Despite success, the model security, as an essential principle, has received increasing attention. Here we consider the worst-case attack, i.e., Byzantine attack (Fang et al. 2024, 2019; Blanchard et al. 2017a), where attackers are omniscient and collusive, and can control several clients to upload arbitrary malicious gradients. Note that Byzantine robustness is especially critical for FR, since in the domains applying FR, e.g., e-commerce, malicious clients can be injected easily by registering new accounts. This raises one question naturally: *With the unique sparse aggregations, how robust the federated recommendation model is against Byzantine attacks?*

For this question, existing Byzantine works cannot be di-

\*These authors contributed equally.

†Corresponding author.

rectly employed, since they mainly focus on dense aggregation in general FL (Rodríguez-Barroso et al. 2022; Xu et al. 2021; Blanchard et al. 2017b). Despite a few prior attacks against FR emerges (Yuan et al. 2023; Yu et al. 2023; Rong et al. 2022; Wu et al. 2022), they also neglect to analyze how the sparse aggregation affects the robustness of FR. We answer this problem by solving two challenges: (1) How to define Byzantine robustness under sparse aggregations? Existing Byzantine attacks and defenses are mainly defined based on the dense aggregation mechanism in the general FL. In FR, due to sparse user-item interaction, for an item, its embedding is updated only by its interacted users, and the remaining users upload zero-valued or empty updates. So the aggregated item embedding may be skewed towards zero-valued when directly applying existing dense aggregators, since the zero-value update is majority. Hence, it is vital to transfer them into FR and re-examine their theoretical guarantee and effectiveness. (2) How to design general Byzantine attacks against FR for attackers with different levels of knowledge and capability in reality. Specifically, it is hard to have full knowledge of all users, due to the large number of participating users in FR. Besides, since user-item interactions are usually sparse, Byzantine clients should not update too many items. Otherwise, a monitor based on the number of user interactions can be triggered easily under such aggressive modifications (Wu et al. 2022).

In this paper, we make the first effort to investigate the Byzantine robustness of federated recommendation from the perspective of sparse aggregations. For the first challenge, we transfer the existing aggregators to FR by treating the aggregation for a single item as the smallest execution unit. Namely, for each item embedding, the gradients are collected and aggregated separately and concurrently. Based on this, we further point out that such a sparse aggregation mechanism of FR will lead to a unique Byzantine vulnerability: items with different degrees receive different amounts of updates, leading to individual robustness. The degrees of all items usually meet long tail distribution in reality (Abdollahpour, Burke, and Mobasher 2019), where most items (named tailed items) are only interacted with by a few users, making them extremely fragile. For the second challenge, we design a series of attack strategies, named **Spattack**, based on the vulnerability from sparse aggregation in FR. Then we categorize them along the attacker’s knowledge and capability into four classes. To be specific, following (Xie, Koyejo, and Gupta 2020; Fang et al. 2019; Baruch, Baruch, and Goldberg 2019), we consider both omniscient attacker (Spattack-O) and limited non-omniscient attacker (Spattack-L). Then we further divide them depending on whether limiting the maximum number of each client’s poisoned items or not. In summary, our contributions are three folds:

- (1) We first systematically study the Byzantine robustness of FR from the perspective of unique sparse aggregation, by treating the aggregation for a single item as the smallest execution unit. We theoretically analyze its convergence guarantee and point out a special vulnerability of FR.
- (2) We propose a family of effective attack strategies, named Spattack, utilizing the vulnerability from sparse aggregation. Then Spattack can be categorized into four different types

along attacker’s knowledge and capability.

- (3) We perform experiments on multiple benchmark datasets for different FR systems. The results show that our Spattack can prevent the convergence of vanilla even defense FR models by only controlling a few malicious clients.

## Background and Preliminary

### Centralized Recommendation

Here, a recommender system contains a set of users  $\mathcal{U} = \{u_1, \dots, u_n\}$  and a set of items  $\mathcal{V} = \{v_1, \dots, v_m\}$ , where  $n$  and  $m$  are the numbers of users and items, respectively. Each user  $u_i \in \mathcal{U}$  has a local training dataset  $\mathcal{D}_i$ , consisting of implicit feedback tuples  $(u_i, v_j, r_{ij})$ . These tuples represent user-item interactions (e.g., purchased, clicked), where  $r_{ij} = 1$  and  $r_{ij} = 0$  indicate positive and negative instances, respectively, i.e., whether  $u_i$  interacted with  $v_j$ . For each user  $u_i$ , we define  $\mathcal{V}_{u_i} = \{v_j \in \mathcal{V} | (u_i, v_j, r_{ij}) \in \mathcal{D}_i\}$  as the set of the items that interact with  $u_i$ . Let  $\mathbf{U} = [u_1, \dots, u_n]$  and  $\mathbf{V} = [v_1, \dots, v_m]$  denote the embeddings of users and items, respectively. The recommender system is trained to predict the rating score  $\hat{r}_{ij} = f_{\Theta}(u_i, v_j)$  between  $u_i$  and  $v_j$ , where  $\hat{r}_{ij}$  represents how much  $u_i$  likes  $v_j$ ,  $f_{\Theta}$  is the score function,  $\{\mathbf{U}, \mathbf{V}, \Theta\}$  are learnable parameters. Then, the system recommends an item list for each user that they might be interested in by sorting the rating scores. In traditional centralized training, the personal dataset  $\mathcal{D}_i$  of each user  $u_i$  is stored on a central server, yielding a total dataset  $\mathcal{D}$  for model training, which will increase the privacy risks.

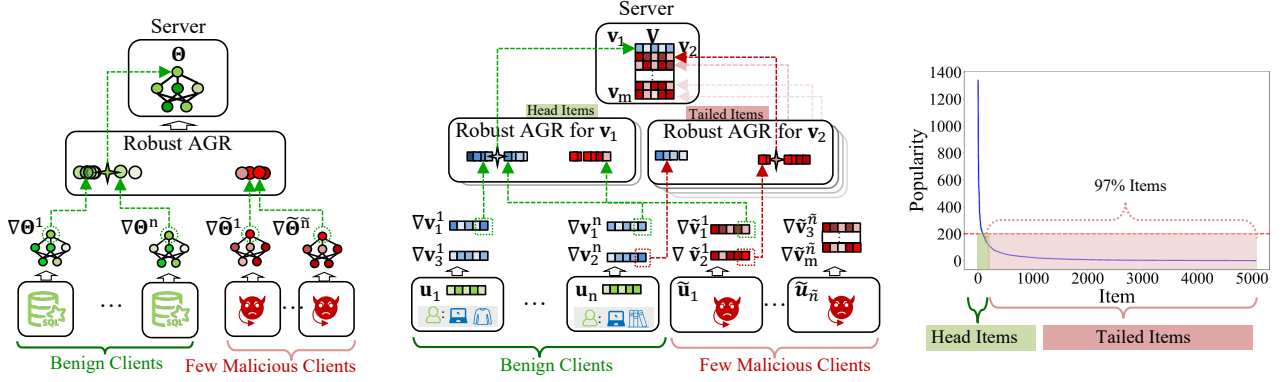
### Federated Recommendation

Considering privacy issues, in FR, the privacy data  $\mathcal{D}_i$  of user  $u_i$  is kept on the local device. The shared model parameters  $\mathbf{V}$  and  $\Theta$  are aggregated over clients by sending the local gradients to a central server. According to the base recommender, the parameters  $\Theta$  are different: in Matrix Factorization (MF) recommender models, the interaction function is fixed and  $\Theta$  is an empty set. In deep learning-based recommender models,  $\Theta$  is the set of weights of neural networks. Following (Rong et al. 2022), we adopt the classic and widely used MF as the base recommender for simplicity, where  $f$  is fixed to be dot product, i.e.,  $\hat{r}_{ij} = u_i \odot v_j$ . Following (Rong et al. 2022), we take Bayesian Personalized Ranking (BPR) (Rendle et al. 2009), a pairwise personalized ranking loss, as the local loss of each client:

$$\mathcal{L}_i(u_i, \mathbf{V}) = - \sum_{\substack{v_j, v_k \in \mathcal{V}_{u_i} \\ r_{ij}=1 \wedge r_{ik}=0}} \ln \sigma(\hat{r}_{ij} - \hat{r}_{ik}), \quad (1)$$

where  $\sigma$  is the logistic sigmoid function. It assumes that the user prefers the positive items over all negative items. In each training iteration, the central server sends the current item embeddings  $\mathbf{V}^t$  to all clients. For each user  $u_i$ , the client computes loss  $\mathcal{L}_i(u_i^t, \mathbf{V}^t)$  then locally updates its private user embedding at epoch  $t$  as follows:

$$u_i^{t+1} \leftarrow u_i^t - \eta \cdot \nabla \mathcal{L}_i^t, \quad (2)$$



(a) Existing Byzantine attack against general FL. (b) Our Byzantine attack against FR. (c) Item popularity of Steam dataset.

Figure 2: Analysis of Byzantine robustness in general FL and FR. Under Byzantine attacks, a robust aggregator can filter outliers in general FL, but fails to defend for tailed items' embedding.

where  $\eta$  is the learning rate. Then  $u_i$  uploads its local item embedding gradients  $\nabla V^{i,t}$  to a central server. After collecting gradients from all clients, the server updates  $V^t$  by:

$$V^{t+1} \leftarrow V^t - \eta \cdot \sum_{i \in [n]} \nabla V^{i,t}. \quad (3)$$

As shown in Fig. 1(b), for each user  $u_i$ , the private interaction history (item list) and user embedding (green  $u_i$  vector) is preserved on the local client device, and only the gradients of item embeddings  $V$  are sent. Throughout the training stage, all users' privacy is well protected.

### Byzantine Attack and Defense

**Byzantine Attack.** In Byzantine attacks, the attacker aims to degrade model performance and even prevent convergence by controlling a few malicious clients. As shown in Fig. 2(a), malicious client  $\tilde{u}_i$  is allowed to send arbitrary (red) gradient  $\nabla \tilde{\Theta}^i$ . Following existing Byzantine attack studies (Xu et al. 2021; Fang et al. 2019; Baruch, Baruch, and Goldberg 2019), considering the worst case, we assume attackers have full knowledge of all benign gradients  $\{\nabla \Theta^1, \dots, \nabla \Theta^n\}$  and all the malicious clients are collusive by default, which help to understand the severity of model poisoning threats.

**Byzantine Defense.** Since servers have no access to the raw training data of clients, the defense is generally implemented on the server side as a robust aggregator, which can filter Byzantine updates and guarantee model convergence. As shown in Fig. 2(a), let  $\{\nabla \Theta^1, \dots, \nabla \Theta^n\}$  be the gradient vectors of  $n$  benign clients in FL. The server collects and aggregates the training gradient of each client model using a federated aggregator. In non-robust FL settings, coordinate-wise Mean in form of  $\text{MEAN}(\nabla \Theta^1, \dots, \nabla \Theta^n) = \frac{1}{n} \sum_{i=1}^n \nabla \Theta^i$  is an effective aggregation rule. However, MEAN can be manipulated by several malicious clients (Blanchard et al. 2017b). Therefore, multiple robust aggregators (Yin et al. 2018; Blanchard et al. 2017b; Xu et al. 2021) are proposed to filter the Byzantine updates. For example, coordinate-wise Median aggregator computes the median for each element  $\Theta_i$  in parameter  $\Theta$  across all clients, yielding 0.5 breakdown point (Yin et al. 2018). Namely, when the fraction of malicious clients is less

than 0.5, the Median aggregator can guarantee the model convergence under Byzantine attacks, yielding the correct gradient (green star) in Fig. 2(a).

### Methodology

In this section, we re-define the Byzantine robustness under sparse aggregation of FR, and theoretically point out the inherent vulnerability. Based on such vulnerability and considering attackers' knowledge and capability, we design a family of attack strategies, named Spattack.

### Problem Definition

For Byzantine attacks, attackers can inject some Byzantine users  $\tilde{U} = \{\tilde{u}_1, \dots, \tilde{u}_{\tilde{n}}\}$ , limiting the proportion of malicious ones less than  $\rho$ , i.e.,  $\tilde{n}/(n + \tilde{n}) < \rho$ . A malicious client  $\tilde{u}_i$  can upload arbitrary gradient values  $\nabla \tilde{V}^{i,t}$  at any epoch  $t$ , to directly perturb the item embedding. The server will collect and aggregate all gradients including benign  $\{\nabla V^{1,t}, \dots, \nabla V^{n,t}\}$  and malicious  $\{\nabla \tilde{V}^{1,t}, \dots, \nabla \tilde{V}^{\tilde{n},t}\}$ . Let  $\text{AGR}(\cdot)$  be the aggregation operator of federated learning, which can be the most common  $\text{MEAN}(\cdot)$  or statistically robust  $\text{MEDIAN}(\cdot)$ . Our Byzantine attacker aims to prevent model convergence, namely, keeping the recommendation loss  $\mathcal{L}_i$  from decreasing. Formally, in FR, the objective of the Byzantine attack is defined as the following optimization problem:

$$\begin{aligned} & \max_{\{\nabla \tilde{V}_i^t : i \in [\tilde{n}]\}} \sum_{i=1}^n (\mathcal{L}_i(\mathbf{u}_i^{t+1}, \mathbf{V}^{t+1}) - \mathcal{L}_i(\mathbf{u}_i^t, \mathbf{V}^t)), \\ & \text{s.t. } \mathbf{V}^{t+1} = \mathbf{V}^t - \eta \cdot \text{AGR}(\{\nabla V^{i,t} : i \in [n]\} \\ & \quad \cup \{\nabla \tilde{V}^{i,t} : i \in [\tilde{n}]\}), \\ & \quad \mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \eta \nabla \mathbf{u}_i^t, \quad \text{for } i \in [n], \\ & \quad \frac{\tilde{n}}{n + \tilde{n}} \leq \rho, \end{aligned} \quad (4)$$

where attackers aim to find the optimal set of malicious gradients  $\{\nabla \tilde{V}_i^t : i \in [\tilde{n}]\}$ , to raise the loss after updating. Before solving this optimization problem, we find that FR has a unique sparse aggregation mechanism defined as follows:

**Definition 1. (Dense/Sparse Aggregation).** Let  $\theta \in \mathbb{R}^d$  be the shared model parameter vector. If there exists an element  $\theta_i$  ( $i \in [d]$ ) for which only a subset of clients can produce valuable updates, the parameter  $\theta$  is sparsely aggregated. If all clients can support it, it is a dense aggregation.

As shown in Fig. 2(a), in general FL, each element (green circle) of model parameters  $\Theta$  is assumed to be involved in all clients’ loss functions, i.e., dense aggregation. Different from FL, for a client  $u_i$  in FR, not all item embeddings  $\mathbf{V} = \{v_1, \dots, v_m\}$  are employed in local loss  $\mathcal{L}_i$  in Eq. 1, i.e., sparse aggregation. For example, client  $u_1$  in Fig. 2(b) only computes valuable gradients  $\{\nabla v_1^1, \nabla v_3^1\}$  for items  $v_1$  and  $v_3$ , while the gradients for the remaining items are either zero or an empty set. When directly applying the aggregators on all  $\mathbf{V}_i$ , the embedding will be skewed towards zero value. Therefore, we need to adapt them to FR.

**Adapting Dense Aggregator to Sparse.** We adapt existing aggregators from dense to sparse aggregation by treating the aggregation for a single item as the smallest execution unit. As shown in Fig. 2(b), the aggregator is conducted separately for each item. Taking the embedding of  $j$ -th item as an example, the embedding is updated by:

$$\mathbf{v}_j^{t+1} = \mathbf{v}_j^t - \eta \cdot \text{AGR}(\{\nabla \mathbf{v}_j^{i,t} \mid \text{user } i \in \mathcal{U}_{v_j}\}), \quad (5)$$

where  $\mathcal{U}_{v_j}$  is the set of users that the item  $v_j$  interacts with,  $\nabla \mathbf{v}_j^{i,t}$  is the gradient of item  $v_j$  sent from client  $u_i$  at epoch  $t$ . Only if the user  $u_i$  has interaction with item  $v_j$ , the gradients  $\nabla \mathbf{v}_j^{i,t}$  can be aggregated to  $\mathbf{v}_j^{t+1}$  separately and concurrently. Intuitively, the numbers of received gradients are varied for different items, leading to each item having personal robustness. Therefore, we need to theoretically re-examine the convergence guarantee of existing aggregators against Byzantine attacks under the sparse aggregation.

## Byzantine Robustness Analysis

**Robustness of FR without Defense.** Like general FL, FR without defense often uses the Mean aggregator to compute the average of input gradients, which is highly susceptible to Byzantine attacks. Even one malicious client can also destroy the Mean aggregator as stated in Proposition 1.

**Proposition 1.** For each item  $v_j$ , let  $\{\nabla \mathbf{v}_j^{i,t} \mid \text{user } i \in \mathcal{U}_{v_j}\}$  be the set of benign gradient vectors at epoch  $t$ . Consider a Mean aggregator averaging updates for each element. Let  $\nabla \tilde{\mathbf{v}}_j$  be a malicious update with arbitrary values in  $\mathbb{R}^d$ . The output of  $\text{MEAN}(\{\nabla \mathbf{v}_j^{i,t} \mid \text{user } i \in \mathcal{U}_{v_j}\} \cup \nabla \tilde{\mathbf{v}}_j) = \frac{1}{|\mathcal{U}_{v_j}|+1} (\sum_{i \in \mathcal{U}_{v_j}} \nabla \mathbf{v}_j^{i,t} + \nabla \tilde{\mathbf{v}}_j)$  can be controlled as zero vector by only single malicious  $\nabla \tilde{\mathbf{v}}_j$ . When all the items are attacked, one malicious client can prevent convergence.

*Proof.* If the attacker registers one malicious client, where the embedding gradient of each item  $v_j$  is  $\nabla \tilde{\mathbf{v}}_j = -\sum_{i \in \mathcal{U}_{v_j}} \nabla \mathbf{v}_j^{i,t}$ , the output of aggregator is zero vector, which can prevent convergence.  $\square$

**Robustness of FR with Defense.** The most common defense method is to use aggregators that are statistically more robust against outliers than Mean. In these defenses, FL

models have a consistently high breakdown point, e.g., when  $\rho < 50\%$ , Median can theoretically guarantee the convergence of FL as proved in (Yin et al. 2018). However, we find that FR models have varied breakdown points for different items, which depends on the item’s degree. Specifically, each item embedding can only be updated by specific clients with whom the item interacts. Obviously, the popular item with massive updates is more robust. Unfortunately, in FR, only a few items interact frequently (head items), while the remaining items interact less frequently (tailed items). We plot the popularity (item degree) of the Steam recommendation dataset (Cheque, Guzmán, and Parra 2019) in Fig. 2(c). We find that 97% tailed items (red long tail area) have interactions less than 200 times, and only 3% head items (green area) interact frequently over 200 times. Therefore, existing statistically-based FL defenses will fail to guarantee the convergence of most items. Formally, let  $x$  be the degree of an item and  $p(x)$  be its probability. We assume that the probability distribution can be defined as a typical power-law distribution  $p(x) = Cx^{-\beta}$ , where  $C$  is a normalization constant and  $\beta$  is the scaling parameter. The failure of defenses can be formally characterized as follows:

**Proposition 2.** Let  $\alpha$  be the breakdown point of robust federated aggregator, the amount of benign and malicious clients are  $n$  and  $\tilde{n}$  respectively, and  $\beta$  is the scaling parameter of the power-law distribution of items’ degree with constant  $C$ . Then at least  $1 - \frac{C}{\beta-1} (\frac{1-\alpha}{\alpha} \tilde{n})^{(1-\beta)}$  percent of items’ embeddings can be broken down.

The proof of Proposition 2 refers to the Appendix. Taking the Steam dataset as an example, the degree distribution of items can be modeled as a typical form of power-law distribution as shown in Fig. 2(c). For example, if an attacker can control  $\rho = 5\%$  clients, each item can receive 197 malicious gradients at most. Clearly, for 97% tailed items that interact less than 200 times, few malicious (red) updates can become the majority and dominate the aggregation. In this case, the statistically robust Median aggregator will pick the majority (red circles), yielding the malicious output (red star). In conclusion, due to the sparse aggregation vulnerability of FR, statistically robust aggregators in FL can also be easily broken down by Byzantine attacker.

## Spattack: Byzantine Attack Strategies

**Intuition.** In Eq. 4, the attacker aims to keep the recommendation loss from decreasing to prevent recommender convergence. Considering the unique vulnerability from sparse aggregation, i.e., the majority of tailed items have a lower breakdown point, we can conclude that: (1) The gradients are farther away from true gradients, the more considerable corruption is. (2) More items are disrupted in the training process, leading to more powerful attacks. Therefore, the attack objective of the proposed Spattack can be simplified to maximally uploading gradients farther away from true gradients and greedily disrupting the embeddings of items.

**Attack Taxonomy.** In real scenarios, depending on the attacker’s knowledge about benign gradients and the maximum number of poisoned items in each malicious client, we

Table 1: Attack Taxonomy. For each malicious client in Spattack, knowledge means knowing benign gradients, and capability refers to poisoning all items.

Spattack	O-D	O-S	L-D	L-S
Knowledge	✓	✓	✗	✗
Capability	✓	✗	✓	✗

outline different scenarios of Spattack that can be launched. As shown in Tab. 1, we have four possible scenarios:

**Spattack-O-D** is considered a worst case, where the attacker is both omniscient and omnipotent, i.e., attackers can obtain benign gradients at each epoch and the maximum number of poisoned items is not limited. Following the first intuition that the malicious gradients farther away from true gradients can cause larger corruption, attackers upload the gradients in the opposite direction of the benign ones. Formally, for a item  $v_j$ , we collect benign gradient  $\nabla v_j^{i,t}$  from  $u_i$ , where  $u_i$  interacts with  $v_j$ , i.e.,  $u_i \in \mathcal{U}_{v_j}$ . Then we compute the sum of the collected benign gradients to obtain the expected gradient  $\nabla \bar{v}_j^t = \sum_{u_i \in \mathcal{U}_{v_j}} \nabla v_j^{i,t}$ . Lastly,

each malicious client  $\tilde{u}_i \in \tilde{\mathcal{U}}$  will upload malicious gradients  $\nabla \tilde{v}_j^{i,t} = -\frac{1}{|\tilde{\mathcal{U}}|} \nabla \bar{v}_j^t$ . Following the second intuition that greedily disrupts items, the attack effectiveness will be maximized by uploading poisoning gradients for all items. In this attack, the non-robust Mean aggregator will output zero gradients, while statistically robust aggregators will select malicious gradients for the majority of tailed items, preventing the convergence of item embeddings. According to Proposition 1 and Proposition 2, even only having a small portion of malicious clients, Spattack-O-D can still guarantee to disrupt the majority of item embeddings.

**Spattack-L-D** uploads random noise as malicious gradients for all items, where attackers are non-omniscient but omnipotent, i.e., attackers do not have any knowledge about the benign gradients but can attack all items. Specifically, attackers construct the malicious gradient by randomly sampling from the Gaussian noise and keeping the same noise in all malicious clients. Under the Mean aggregator, the aggregated gradients can be skewed by such noise. Even worse, the statistically robust aggregators, e.g., Median, can pick the uploaded random noise as output for tailed items. So this attack can still prevent model convergence.

**Spattack-O-S and Spattack-L-S** only upload malicious gradients for partial items, where attackers are non-omnipotent. Let  $\tilde{m}_{max}$  be the maximum number of poisoned items in each malicious client. The larger  $\tilde{m}_{max}$ , the stronger the attack, but the excessive  $\tilde{m}_{max}$  may lead to the attack being detected. To limit malicious users to behaving like benign users, we restrict  $\tilde{m}_{max}$  as the maximum number of interactions in benign clients. Specifically, to make the injections of malicious clients as imperceptible and effective as possible, based on the distribution of item popularity, we use a sampling operation to determine the poisoned items for each malicious client. Therefore, the attacker can automatically assign more malicious gradients to the items having more interactions. Then we generate malicious gradients based on the opposite benign gradients (Spattack-O-S) or random noise (Spattack-L-S), respectively.

Table 2: Statistics of datasets.

Dataset	#Users	#Items	#Edges	Sparsity
ML100K	943	1,682	100,000	93.70 %
ML1M	6,040	3,706	1,000,209	95.53 %
Steam	3,753	5,134	114,713	99.40 %

## Experiment

We conduct extensive experiments to answer the following research questions. **RQ1:** How does Spattack perform compared with existing Byzantine attacks? **RQ2:** Can Spattack break the defenses deployed on FR? **RQ3:** Can Spattack transfer to different FR systems? **RQ4:** How do hyperparameters impact on Spattack? Given the limited space, please refer to the Appendix for more detailed experiments.

### Experimental Setup

**Datasets and Federated Recommender Systems.** Following (Rong et al. 2022), Spattack is evaluated on three widely used datasets, including movie recommendation datasets ML1M and ML100K (Harper and Konstan 2016), and game recommendation dataset Steam (Cheuque, Guzmán, and Parra 2019). The dataset statistics refer to Tab. 2. The test set is divided with the leave-one-out method, where the latest interaction of a user is left as the test set and the remaining interactions as the training set. FedMF (Rong et al. 2022) and the SOTA FedGNN (Wu et al. 2021) are selected as evaluation models. More dataset and reproducibility details are in the Appendix.

**Evaluation Protocols.** We utilize two common evaluation protocols, including hit ratio (HR) and normalized discounted cumulative gain (nDCG) at ranks 5 and 10. For each user, since ranking the test item among all items is time-consuming, following the widely-used strategy (He et al. 2017), we randomly sample 100 items that do not interact with the user, then rank the test item among the 100 items. Notably, all metrics are only calculated on benign clients.

**Baselines.** We compare Spattack with two categories of methods. First is the data poisoning attack, where attackers generate malicious gradients by modifying training data. LabelFlip (Tolpegin et al. 2020) flips training labels for poisoning, while FedAttack (Wu et al. 2022) uses misaligned samples. Second is model poisoning attacks, where attackers directly modify the uploaded gradients. Gaussian (Fang et al. 2019) estimates the Gaussian distribution of benign gradients and then samples from it. LIE (Baruch, Baruch, and Goldberg 2019) adds small amounts of noise towards the average of benign gradients. Cluster (Yu et al. 2023) uploads malicious gradients that aim to make item embeddings collapse into several dense clusters. Fang (Fang et al. 2019) adds noise to opposite directions of the average normal gradient. More details can be found in the Appendix.

**Byzantine Defense Strategies.** We evaluate Spattack performance under the following defense strategies: Mean is the vanilla non-robust aggregator that computes the mean value of gradients for each dimension. Median (Yin et al. 2018) is a statistically robust aggregator with a 0.5 breakdown point, computing the element-wise median value. Trimmed-mean (Yin et al. 2018) trims several extreme values for each dimension and then averages the rest. Krum (Blanchard et al.

Table 3: Comparison of Spattack with baselines under a 3% malicious rate. Lower scores represent better attack effectiveness. We additionally report the performance drop (%) compared with the performance on the clean model.

Dataset	Metric	Clean	LabelFlip	FedAttack	Gaussian	LIE	Cluster	Fang	Type L-S	Type L-D	Type O-S	Type O-D
ML100K	HR@5	0.2513	0.2517 (-3%)	0.2550 (-2%)	0.2550 (-2%)	0.2539 (-2%)	0.2461 (-5%)	0.1957 (-25%)	0.1018 (-59%)	0.0721 (-71%)	0.0594 (-76%)	<b>0.0530</b> (-79%)
	nDCG@5	0.1643	0.1706 (-3%)	0.1721 (-2%)	0.1729 (-1%)	0.1724 (-2%)	0.1678 (-4%)	0.1229 (-30%)	0.0620 (-62%)	0.0380 (-77%)	0.0362 (-78%)	<b>0.0339</b> (-79%)
	HR@10	0.4051	0.4083 (-2%)	0.4094 (-2%)	0.4116 (-2%)	0.4116 (-2%)	0.3982 (-5%)	0.2919 (-30%)	0.2163 (-47%)	0.1601 (-60%)	0.0997 (-75%)	<b>0.0944</b> (-77%)
	nDCG@10	0.2131	0.2206 (-2%)	0.2213 (-2%)	0.2230 (-1%)	0.2229 (-1%)	0.2166 (-4%)	0.1541 (-32%)	0.0980 (-54%)	0.0658 (-69%)	0.0492 (-77%)	<b>0.0470</b> (-78%)
ML1M	HR@5	0.3121	0.3051 (-1%)	0.3056 (-1%)	0.3053 (-1%)	0.3054 (-1%)	0.3033 (-2%)	0.2827 (-9%)	0.1007 (-68%)	0.0921 (-71%)	0.0925 (-70%)	<b>0.0907</b> (-71%)
	nDCG@5	0.2054	0.2013 (-2%)	0.2021 (-1%)	0.2017 (-1%)	0.2018 (-1%)	0.2004 (-2%)	0.1858 (-9%)	0.0581 (-72%)	<b>0.0521</b> (-75%)	0.0553 (-73%)	0.0549 (-73%)
	HR@10	0.4626	0.4632 (-1%)	0.4634 (-1%)	0.4634 (-1%)	0.4634 (-1%)	0.4592 (-2%)	0.3977 (-15%)	0.2141 (-54%)	0.1935 (-58%)	0.1753 (-62%)	<b>0.1679</b> (-64%)
	nDCG@10	0.2539	0.2522 (-1%)	0.2528 (-1%)	0.2526 (-1%)	0.2526 (-1%)	0.2506 (-2%)	0.2231 (-12%)	0.0939 (-63%)	0.0846 (-67%)	0.0817 (-68%)	<b>0.0793</b> (-69%)
Steam	HR@5	0.5729	0.4792 (-15%)	0.4798 (-15%)	0.4879 (-14%)	0.4862 (-14%)	0.4263 (-25%)	0.0278 (-95%)	0.0426 (-93%)	<b>0.0139</b> (-98%)	0.0671 (-88%)	0.0685 (-88%)
	nDCG@5	0.3815	0.3157 (-17%)	0.3172 (-16%)	0.3216 (-15%)	0.3209 (-15%)	0.2750 (-27%)	0.0160 (-96%)	0.0261 (-93%)	<b>0.0080</b> (-98%)	0.0390 (-90%)	0.0408 (-89%)
	HR@10	0.6933	0.6429 (-7%)	0.6431 (-7%)	0.6474 (-6%)	0.6471 (-6%)	0.6220 (-10%)	0.0619 (-91%)	0.0834 (-88%)	<b>0.0322</b> (-95%)	0.1308 (-81%)	0.1287 (-81%)
	nDCG@10	0.4207	0.3685 (-12%)	0.3700 (-12%)	0.3732 (-11%)	0.3730 (-11%)	0.3386 (-19%)	0.0269 (-94%)	0.0391 (-91%)	<b>0.0138</b> (-97%)	0.0593 (-86%)	0.0601 (-86%)

2017b) picks the gradient that is the most similar to other uploaded gradients. Norm (Suresh et al. 2019) clips the norm of gradients with a given threshold.

### Attack Performance Evaluation (RQ1)

We compare the proposed Spattack against existing SOTA attack baselines under 3% malicious ratio. The experimental results are reported in Tab. 3, we find:

- Spattack can prevent FR convergence by controlling a few malicious clients. For example, Spattack can achieve a 47%-98% performance drop under 3% malicious clients, demonstrating that FR is extremely vulnerable to Spattack.
- Spattack significantly outperforms other baselines. The explanation is that Spattack fully utilizes the sparse aggregation vulnerability by greedily breaking more items. Specifically, LabelFlip and FedAttack only indirectly manipulate the gradient by modifying data, while LIE, Cluster and Fang directly manipulate the gradients and thus can achieve higher attack impacts. Although Fang also perturbs in opposite directions of benign gradients, the malicious gradients are skewed to zero vector without considering the sparse aggregation of FR, leading to less effective attacks.
- The results on Steam overall drop more than ML100K and ML1M. A possible reason is that Steam involves fewer interactions on average (referring to the sparsity in Tab. 2), meaning there are more tailed items, which makes the model more susceptible to attacks.

### Attack Effectiveness under Defense (RQ2)

We also evaluate the effectiveness of Spattack under different defenses. We set malicious ratio  $\rho$  as 1%, 3% and 5% for omniscient Spattack-O, and set the higher 5%, 10% and 15% for the harder non-omniscient Spattack-L. We equip FR with

Mean, Median and Norm aggregators for all attacks. Since TrimM and Krum assume the number of malicious updates is fixed for each item, but Spattack-O/L-S uploads different numbers of updates for each item, making them cannot be applied. As shown in Fig. 3, more results and analysis are in the Appendix. We have observations as follows:

- With only 5% malicious clients, Spattack can dramatically degrade recommendation performance and even prevent convergence. The explanation is that different items have varied amounts of updates, and the defense of tailed items can be more easily broken than head items.
- When the attacker’s knowledge and capability are limited, with the increasing malicious ratio  $\rho$ , the performance of defense FR consistently decreases even reaching an untrained model. The results also demonstrate that hiding the gradients of benign clients cannot protect FR, because the attacker can break the defense using only random noise.

### Transferability of Attack (RQ3)

To demonstrate the generalizability of Spattack to other federated recommender systems, we perform Spattack on the SOTA FedGNN (Wu et al. 2021) by extra uploading the malicious gradients of the GNN model. No Defense and Defense correspond to mean and median aggregators, respectively. The malicious ratio  $\rho$  is set to 10%. Please refer to the Appendix for more results and analysis. As shown in Fig. 4, we have the following observations:

- The performance of FedGNN dramatically drops under Spattack, demonstrating the common vulnerability of FedMF and FedGNN. Even though the parameters of GNN are densely aggregated, the attacker can still prevent convergence of model training by poisoning item embeddings.
- Spattack can achieve more effective attacks under defense.

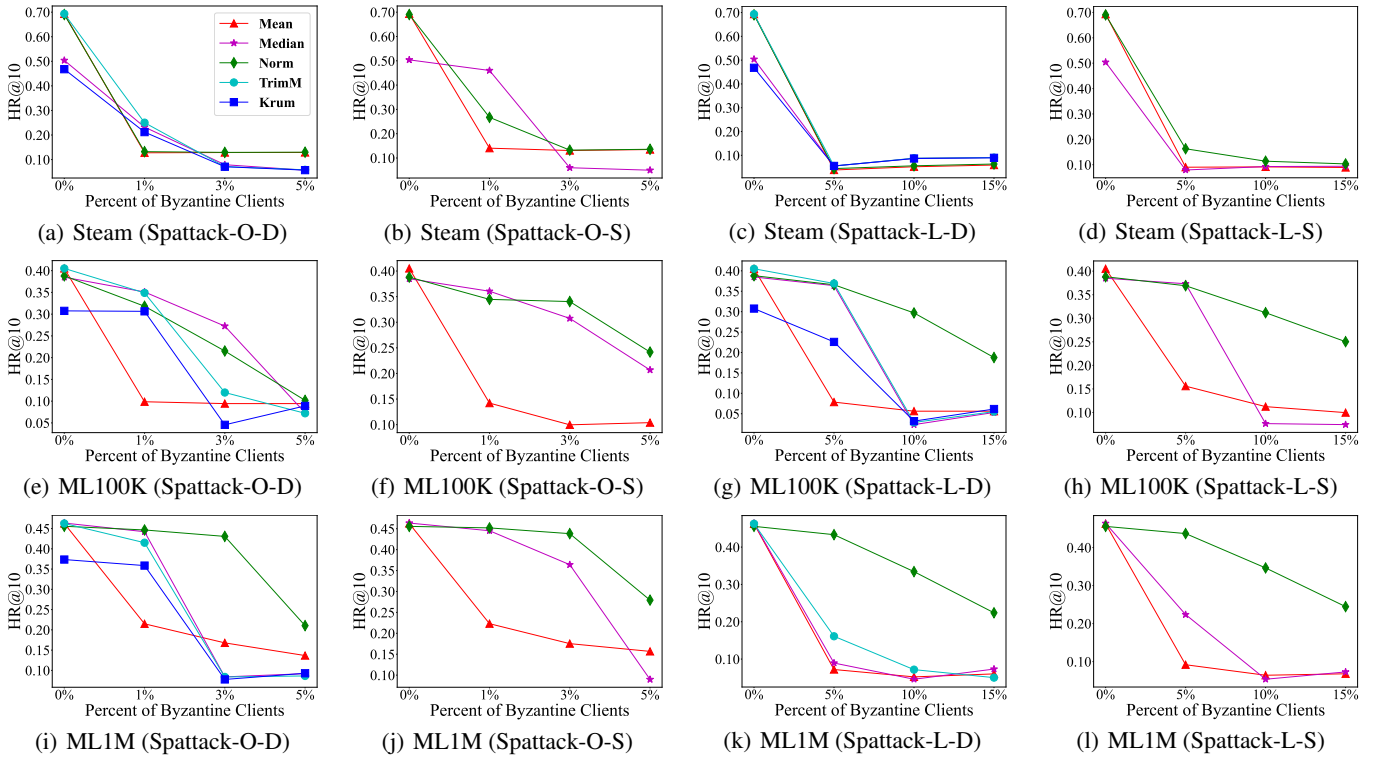


Figure 3: Performance of Spattack against multiple defense strategies under different ratios of Byzantine clients.

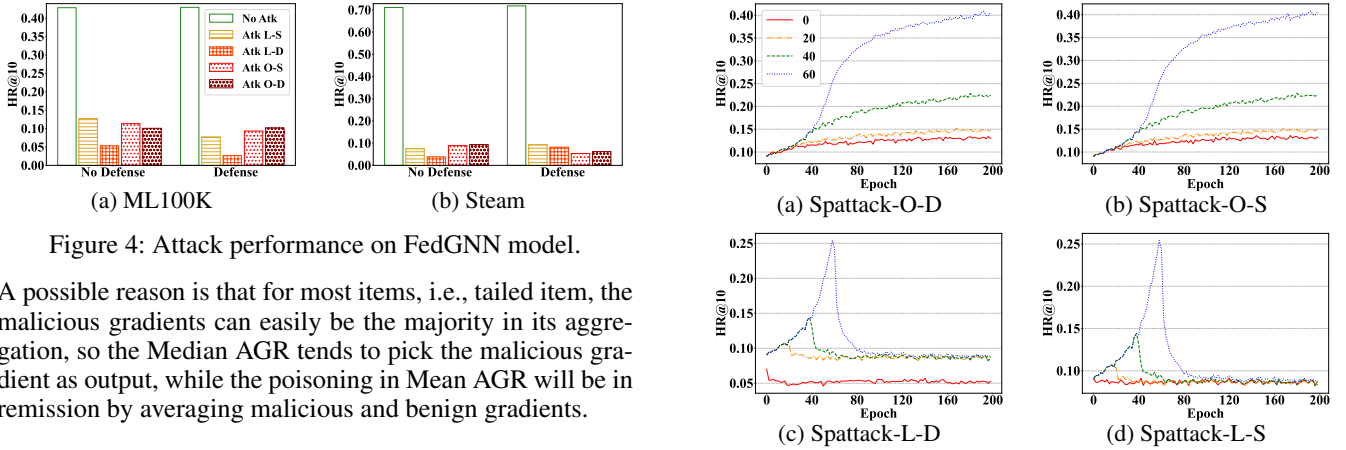


Figure 4: Attack performance on FedGNN model.

A possible reason is that for most items, i.e., tailed item, the malicious gradients can easily be the majority in its aggregation, so the Median AGR tends to pick the malicious gradient as output, while the poisoning in Mean AGR will be in remission by averaging malicious and benign gradients.

### Hyperparameter Analysis (RQ4)

Lastly, we investigate the impact of the hyper-parameter on Spattack. In Fig. 5, we show the convergence of FR under mean aggregators on Steam, where the malicious ratio is set to 10%, and the results correspond to starting attacks at 0, 20, 40 and 60. Please refer to the Appendix for more results and analysis. We have the following observations:

- Spattack with a small starting epoch tends to have better attack performance because the model has converged under a large starting epoch.
- When Spattack is launched, Spattack-O prevents the model from continuing to converge, while Spattack-L causes the performance dramatically drops. The reason is that Spattack-O uploads malicious gradients with an average equal to the negative of the benign gradients' average, resulting in zero gradients after aggregation.

Figure 5: Attack performance on different start epochs.

## Conclusion

In this paper, we first systematically study the Byzantine robustness of federated recommender from the perspective of sparse aggregation, where the item embedding in FR can only be updated by partial clients, instead of full clients (dense aggregation in general FL). Then we design a series of attack strategies, called Spattack, based on the vulnerability from sparse aggregation in FR. Our Spattack can be employed by attackers with different levels of knowledge and capability. Extensive experimental results demonstrated that FR is extremely fragile to Spattack. In the future, we aim to design a more robust aggregator in FR from the perspective of sparse aggregation, which focuses on the robust aggregation for tailed items.

## Acknowledgments

This work is supported by the National Key Research and Development Program of China (2023YFF0725103), the National Natural Science Foundation of China (U22B2038, 62322203, 62172052, 62192784), and the Young Elite Scientists Sponsorship Program (No.2023QNRC001) by CAST.

## References

- Abdollahpouri, H.; Burke, R.; and Mobasher, B. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-ranking. In *The Florida AI Research Society*.
- Ammaduddin, M.; Ivannikova, E.; Khan, S. A.; et al. 2019. Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System. *arXiv: Information Retrieval*.
- Baruch, G.; Baruch, M.; and Goldberg, Y. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *Neural Information Processing Systems*.
- Blanchard, P.; Mhamdi, E. M. E.; Guerraoui, R.; et al. 2017a. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NIPS*.
- Blanchard, P.; Mhamdi, E. M. E.; Guerraoui, R.; et al. 2017b. Machine learning with adversaries: byzantine tolerant gradient descent. *Neural Information Processing Systems*.
- Chai, D.; Wang, L.; Chen, K.; et al. 2019. Secure Federated Matrix Factorization. *IEEE Intelligent Systems*.
- Chen, C.; Zhang, J.; Tung, A. K.; Kankanhalli, M.; and Chen, G. 2020. Robust federated recommendation system. *arXiv preprint arXiv:2006.08259*.
- Cheque, G.; Guzmán, J.; and Parra, D. 2019. Recommender Systems for Online Video Game Platforms: the Case of STEAM. *Companion Proceedings of The 2019 World Wide Web Conference*.
- Elkahky, A. M.; Song, Y.; and He, X. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. *Proceedings of the 24th International Conference on World Wide Web*.
- Fan, W.; Ma, Y.; Yin, D.; et al. 2019. Deep social collaborative filtering. *Proceedings of the 13th ACM Conference on Recommender Systems*.
- Fang, M.; Cao, X.; Jia, J.; et al. 2019. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *USENIX Security Symposium*.
- Fang, M.; Zhang, Z.; Hairi; Khanduri, P.; Liu, J.; Lu, S.; Liu, Y.; and Gong, N. 2024. Byzantine-robust decentralized federated learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2874–2888.
- Fu, S.; Xie, C.; Li, B.; et al. 2019. Attack-Resistant Federated Learning with Residual-based Reweighting. *ArXiv*.
- Harper, F. M.; and Konstan, J. A. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*
- He, X.; Liao, L.; Zhang, H.; et al. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*.
- Liu, Z.; Yang, L.; Fan, Z.; et al. 2021. Federated Social Recommendation with Graph Neural Network. *ACM Transactions on Intelligent Systems and Technology (TIST)*.
- Luo, S.; Xiao, Y.; and Song, L. 2022. Personalized Federated Recommendation via Joint Representation Learning, User Clustering, and Model Adaptation. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*.
- Lyu, L.; Yu, H.; Ma, X.; et al. 2022. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*.
- Lyu, L.; Yu, H.; Zhao, J.; et al. 2020. Threats to federated learning. *Federated Learning: Privacy and Incentive*.
- Ma, H.; Yang, H.; Lyu, M. R.; et al. 2008. SoRec: social recommendation using probabilistic matrix factorization. In *International Conference on Information and Knowledge Management*.
- McMahan, H. B.; Moore, E.; Ramage, D.; et al. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics*.
- Mhamdi, E. M. E.; Guerraoui, R.; and Rouault, S. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *International Conference on Machine Learning*.
- Pillutla, K.; Kakade, S. M.; and Harchaoui, Z. 2019. Robust Aggregation for Federated Learning. *IEEE Transactions on Signal Processing*.
- Rendle, S.; Freudenthaler, C.; Gantner, Z.; et al. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. *ArXiv*.
- Rodríguez-Barroso, N.; López, D. J.; Luzón, M. V.; et al. 2022. Survey on Federated Learning Threats: concepts, taxonomy on attacks and defences, experimental study and challenges. *ArXiv*.
- Rong, D.; Ye, S.; Zhao, R.; et al. 2022. FedRecAttack: Model Poisoning Attack to Federated Recommendation. *2022 IEEE 38th International Conference on Data Engineering (ICDE)*.
- Sun, Z.; Xu, Y.; Liu, Y.; He, W.; Kong, L.; Wu, F.; Jiang, Y.; and Cui, L. 2024. A survey on federated recommendation systems. *IEEE Transactions on Neural Networks and Learning Systems*.
- Suresh, A. T.; McMahan, B.; Kairouz, P.; et al. 2019. Can You Really Backdoor Federated Learning. *arXiv: Learning*.
- Tolpegin, V.; Truex, S.; Gursoy, M. E.; et al. 2020. Data Poisoning Attacks Against Federated Learning Systems. *Cornell University - arXiv*.
- Wang, H.; Zhang, F.; Xie, X.; et al. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. *Proceedings of the 2018 World Wide Web Conference*.
- Wu, C.; Wu, F.; An, M.; et al. 2019a. NPA: Neural News Recommendation with Personalized Attention. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.



- Wu, C.; Wu, F.; Lyu, L.; et al. 2021. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*.
- Wu, C.; Wu, F.; Qi, T.; et al. 2022. FedAttack: Effective and Covert Poisoning Attack on Federated Recommendation via Hard Sampling. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- Wu, Y.; Hu, X.; Sun, Y.; Zhou, Y.; Zhu, W.; Rao, F.; Schiele, B.; and Yang, X. 2024a. Number it: Temporal Grounding Videos like Flipping Manga. *arXiv preprint arXiv:2411.10332*.
- Wu, Y.; Zhou, S.; Yang, M.; Wang, L.; Zhu, W.; Chang, H.; Zhou, X.; and Yang, X. 2024b. Unlearning Concepts in Diffusion Model via Concept Domain Correction and Concept Preserving Gradient. *arXiv preprint arXiv:2405.15304*.
- Wu, Z.; Ling, Q.; Chen, T.; et al. 2019b. Federated Variance-Reduced Stochastic Gradient Descent With Robustness to Byzantine Attacks. *IEEE Transactions on Signal Processing*.
- Xie, C.; Koyejo, O.; and Gupta, I. 2020. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, 261–270. PMLR.
- Xu, J.; Huang, S.-L.; Song, L.; et al. 2021. Byzantine-robust Federated Learning through Collaborative Malicious Gradient Filtering. *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*.
- Yan, B. 2024. Federated Graph Condensation with Information Bottleneck Principles. *arXiv preprint arXiv:2405.03911*.
- Yan, B.; Cao, Y.; Wang, H.; Yang, W.; Du, J.; and Shi, C. 2024. Federated heterogeneous graph neural network for privacy-preserving recommendation. In *Proceedings of the ACM on Web Conference 2024*, 3919–3929.
- Yin, D.; Chen, Y.; Ramchandran, K.; et al. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *arXiv: Learning*.
- Ying, R.; He, R.; Chen, K.; et al. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
- Ying, S. 2020. Shared MF: A privacy-preserving recommendation system. *ArXiv*.
- Yu, Y.; Liu, Q.; Wu, L.; et al. 2023. Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Yuan, W.; Nguyen, Q. V. H.; He, T.; et al. 2023. Manipulating Federated Recommender Systems: Poisoning with Synthetic Users and Its Countermeasures. In *SIGIR*.
- Zhang, H.; Luo, F.; Wu, J.; He, X.; and Li, Y. 2023a. LightFR: Lightweight federated recommendation with privacy-preserving matrix factorization. *ACM Transactions on Information Systems*.
- Zhang, M.; Wang, X.; Shi, C.; Lyu, L.; Yang, T.; and Du, J. 2023b. Minimum topology attacks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, 630–640.
- Zhang, M.; Wang, X.; Zhu, M.; Shi, C.; Zhang, Z.; and Zhou, J. 2022. Robust heterogeneous graph neural networks against adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4, 4363–4370.
- Zhang, S.; Yin, H.; Chen, T.; et al. 2021. PipAttack: Poisoning Federated Recommender Systems for Manipulating Item Promotion. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*.
- Zhang, Z.; Wang, X.; Zhou, H.; Yu, Y.; Zhang, M.; Yang, C.; and Shi, C. 2024a. Can Large Language Models Improve the Adversarial Robustness of Graph Neural Networks? *arXiv preprint arXiv:2408.08685*.
- Zhang, Z.; Zhang, M.; Yu, Y.; Yang, C.; Liu, J.; and Shi, C. 2024b. Endowing Pre-trained Graph Models with Provable Fairness. In *Proceedings of the ACM on Web Conference 2024*, 1045–1056.

## Related Work

**Federated Recommender.** Federated learning aims to collaboratively train a shared model based on the distributed data in a privacy-preserving manner (Yan et al. 2024; McMahan et al. 2016; Zhang et al. 2023a). Accordingly, a federated recommender ensures individual users’ historical data is locally stored and only uploads intermediate data to the server for collaborative training. In this process, the user’s rating behaviors (the set of interacted items or rating scores) are private information. Ammaduddin et al. (2019) first proposed a federated collaborative filter framework for the privacy-preserving recommendation. The user embeddings are stored and updated locally while the gradients of item embeddings are uploaded to the server for aggregation. Moreover, for better privacy, (Chai et al. 2019) applied homomorphic encryption; (Ying 2020) further improved the efficiency by utilizing secret sharing instead of homomorphic encryption. Recently, federated recommendations based on graph neural networks have emerged (Wu et al. 2021; Liu et al. 2021; Luo, Xiao, and Song 2022; Yan 2024; Zhang et al. 2023b), further incorporating high-order user-item interactions into local training data. Overall, most existing federated recommender systems follow the paradigm where the gradients of item embeddings are uploaded to the server for aggregation. So they are all sparse aggregations where each item embedding can only be updated by partial clients, leading to varied robustness of each item. This paper sheds the first light on this unique vulnerability in the view of sparse aggregation.

**Robustness of Federated Learning.** The security of federated learning has drawn increasing attention in recent years (Lyu et al. 2022, 2020; Zhang et al. 2024a, 2022, 2024b; Wu et al. 2024b,a), and a large number of attacks against FL have been proposed. Among the attack strategies, the Byzantine attack is one of the most popular attacks (Rodríguez-Barroso et al. 2022). The classic mean aggregator can be easily skewed by arbitrary updates from Byzantine clients. Therefore, a lot of Byzantine defense based on statistics has been proposed in recent year (Yin et al. 2018; Blanchard et al. 2017b; Mhamdi, Guerraoui, and Rouault 2018; Xu et al. 2021; Rodríguez-Barroso et al. 2022; Pillutla, Kakade, and Harchaoui 2019; Wu et al. 2019b; Fu et al. 2019; Chen et al. 2020), which aimed to filter the Byzantine updates and guarantee the convergence of federated learning. Although the Byzantine robustness problem is well-studied in FL, existing Byzantine attacks and defenses of FL are defined based on the dense aggregation and cannot apply to our sparse aggregation in FR. More recently, a few attacks have been proposed against federated recommender, among which, (Rong et al. 2022; Zhang et al. 2021) focus on targeted attacks aiming to promote target items by increasing their exposure chances, and (Wu et al. 2022) employs improper positive/negative samples to manipulate model parameters indirectly. (Yu et al. 2023) uploads poisonous gradients that collapse all item embeddings to several clusters to confuse different items. However, they all neglect the unique sparsity in federated recommender.

## Notations

We present all notations relevant to our paper in Tab. 4.

Table 4: Notations

$\mathcal{D}$	all interactions
$\mathcal{D}_i$	local interaction of user $u_i$
$\mathcal{U}$	set of $n$ benign users
$\tilde{\mathcal{U}}$	set of $\tilde{n}$ malicious users
$\mathcal{V}$	set of $m$ items
$\mathcal{V}_{u_i}$	set of items interacting with $u_i$
$\mathcal{U}_{v_j}$	set of users interacting with $v_j$
$\mathbf{U}$	the user embeddings where $\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_n\}$
$\mathbf{V}$	the item embeddings where $\mathbf{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_m\}$
$\nabla \mathbf{V}^{i,t}$	the embedding gradient of $\mathcal{V}$ from benign user $u_i$ at epoch $t$
$\nabla \tilde{\mathbf{V}}^{i,t}$	the embedding gradient of $\mathcal{V}$ from malicious user $\tilde{u}_i$ at epoch $t$
$\nabla \mathbf{v}_j^{i,t}$	the embedding gradient of $v_j$ from benign user $u_i$ at epoch $t$
$\nabla \tilde{\mathbf{v}}_j^{i,t}$	the embedding gradient of $v_j$ from malicious user $\tilde{u}_i$ at epoch $t$
$\Theta$	the parameters of neural network model
$[n]$	Set of integers $\{1, \dots, n\}$
$\eta$	learning rate
$\rho$	the proportion of malicious users
$\alpha$	the breaking point of statistically robust aggregator

### Detailed Proof for Proposition 2

*Proof.* Consider the robust aggregator with breaking point  $\alpha$ , the model convergence can be guaranteed when the number of malicious clients meets  $\frac{\tilde{n}}{n+\tilde{n}} < \alpha$  in FL. While in FR, given an item  $v$  with degree  $d_v$ , the aggregator only collects  $d_v$  benign gradients in sparse aggregation. In Byzantine attacks, all the malicious clients can easily collude to send consistent malicious gradients to certain item  $v$ . Once  $\tilde{n}/(d_v + \tilde{n}) > \alpha$ , namely  $v$ ’s degree meets  $d_v < (\tilde{n} - \alpha\tilde{n})/\alpha$ , the malicious updates will ultimately dominate and hence the statically robust aggregator will be tricked to pick the malicious updates. Let  $p(x) = Cx^{-\beta}$  be the power-law distribution of items’ degrees. Its cumulative distribution function  $P(x)$  is defined as the probability that the quantity of the item’s degree is larger than  $x$ :

$$\begin{aligned} P(x) &= \Pr(X > x) = C \int_x^{+\infty} p(X) dX \\ &= C \int_x^{+\infty} X^{-\beta} dX = \frac{Cx^{(1-\beta)}}{\beta-1}. \end{aligned} \quad (6)$$

So the probability that item’s degree is smaller than  $(\tilde{n} - \alpha\tilde{n})/\alpha$  can be calculated as following:

$$1 - P(X > \frac{1-\alpha}{\alpha}\tilde{n}) = 1 - \frac{C}{\beta-1} (\frac{1-\alpha}{\alpha}\tilde{n})^{(1-\beta)}. \quad (7)$$

□

### Reproducibility Supplement

#### Dataset and Evaluation Metric

Following (Rong et al. 2022), Spattack is evaluated on three widely used datasets, including a movie recommendation dataset MovieLens-1M (ML1M) (Harper

and Konstan 2016), the small version MovieLens-100K (ML100K), and game recommendation dataset Steam-200K (Steam) (Cheuque, Guzmán, and Parra 2019). For all datasets, we closely follow the dataset configurations in previous works (He et al. 2017; Rong et al. 2022) by unifying interactions as implicit feedback and removing duplicate interactions. To evaluate the ranking quality of the test item recommendation, we adopt two common evaluation protocols, i.e., hit ratio (HR) and normalized discounted cumulative gain at rank  $K$  (nDCG@ $K$ ).

Here, we use  $K$  as 5 and 10. For each user, since ranking the test item among all items is time-consuming, following the widely-used strategy (Elkahky, Song, and He 2015; He et al. 2017), we randomly sample 100 items from the items that have not interacted with the user, then rank the test item among the 100 items. The HR@ $K$  indicates whether the test item is ranked in the top- $K$ , and the nDCG@ $K$  takes position significance into account. The higher HR@ $K$  and nDCG@ $K$  indicate better recommendation performance. Note that these metrics are only calculated on benign clients.

### Federated Recommender Systems

We use FedMF (Rong et al. 2022) as our target model’s architecture in evaluation. We outline the details for reproducibility below. Following (Rong et al. 2022), with BPR loss, the FedMF updates user embeddings locally and optimizes item embeddings on the server. We set the unit size of embeddings to 32. To evaluate the generalization ability of Spattack, we also conduct attacks on the state-of-the-art FedGNN (Wu et al. 2021), which can collaboratively train GNN models meanwhile exploiting high-order user-item interaction information with privacy well protected. We use BPR loss to train a 2-layer FedGNN model, where both item embeddings and GNN parameters are globally optimized. For the input user and item embedding, the dimension is set to 32. For the hidden layer, we set the hidden unit size to 64. Stochastic gradient descent is selected as the default optimization algorithm, and its learning rate is 0.01.

### Baselines

- LabelFlip (Tolpegin et al. 2020) poisons data by flipping the training labels of malicious clients and does not require knowledge of the training data distribution. Each malicious client uses positive samples as negative samples and uses negative samples as positive samples.
- FedAttack (Wu et al. 2022) conducts data poisoning by employing improper positive/negative samples. Each malicious client selects the items that are most similar to the user’s interest as negative samples while regarding items that are most dissimilar to the user’s interest as positive samples.
- Gaussian (Fang et al. 2019) estimates a Gaussian distribution of benign gradients and then uploads samples from it.
- LIE (Baruch, Baruch, and Goldberg 2019) adds small amounts of noise towards the average of benign gradients. We assign 0.1 as the scaling factor that affects the standard deviation of model parameters.
- Fang (Fang et al. 2019) adds noise to opposite directions of the average normal gradient. We select the attack scaling factor from randomly uniform samples from [3, 4].

Table 5: The number of malicious clients under different attack ratios  $\rho$ .

Dataset	1%	3%	5%	10%	15%
ML100K	9	29	49	105	166
ML1M	61	186	317	671	1066
Steam	37	116	197	417	662

- Cluster (Yu et al. 2023) uploads malicious gradients that aim to make item embeddings collapse into several dense clusters. We set the initial number of clusters as 1. The range of the number of clusters and the threshold is set to [1, 10].

### Additional Implementation Details

- **General settings.** The default optimization algorithm is stochastic gradient descent, and the learning rate is 0.01. We set the epoch number to 200 and the malicious clients conduct attacks from the first epoch. The number of malicious clients under different attack ratios  $\rho$  are shown in Tab. 5.
- **FR models.** For the input user and item embedding, the dimension is set to 32. The hidden layer unit size is 64 in FedGNN. In FedMF, following (Rong et al. 2022), we initialize the representations of users and items with normal distribution with a mean value of 0 and a standard deviation of 0.01. In FedGNN, we initialize the GNN weight matrices with Xavier Glorot’s initialization with a gain value of 1.
- **Spattack-O-S/D.** For omniscient attackers, we generate malicious gradients in the opposite direction of the benign ones. Such that the mean aggregator will output zero gradients, preventing the convergence of item embeddings.
- **Spattack-L-S/D.** For non-omniscient attackers, we sample the Gaussian noise with a mean value of 0 and a standard deviation of 1 as the current gradient. And all malicious clients will upload the same malicious gradient.
- **Spattack-O/L-S.** When the maximum number of poisoned items  $\tilde{n}_{max}$  is limited, we sample the poisoned item list for each malicious client from the distribution of item degree. The items with more benign updates will receive more malicious updates. And the sampling operation of the malicious clients is non-repeatable.

### Experiment Environment and Source Code

All experiments are conducted on a Linux server with one GPU (NVIDIA GeForce RTX 3090 GPU) and CPU (Intel Xeon Gold 6348), and its operating system is Ubuntu 18.04.5. We implement Spattack with the deep learning library PyTorch. The main source code of Spattack can be found at <https://github.com/zhongjian-zhang/Spattack>.

## Supplemental Experimental Results

### Performance Evaluation of Spattack-O-D

We report the detailed results of Spattack-O-D in Tab. 6, where attackers know total benign gradients and with no limitation of maximum number of updating items. We first find that with the increasing malicious ratio  $\rho$ , the defense performance under Spattack-O-D consistently decreases and even reaches an untrained state, indicating existing defenses are more fragile in FR than expected. When  $\rho = 3\%$ , the averaged performance drop rate of defenses is about 71%, and

Table 6: Recommendation performance under Spattack-O-D. We also report the performance drop rate w.r.t. clean model.

Dataset	Defense	1%				3%				5%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0551 (-78%)	0.0354 (-78%)	0.0986 (-76%)	0.0491 (-77%)	0.0530 (-79%)	0.0339 (-79%)	0.0944 (-77%)	0.0470 (-78%)	0.0573 (-77%)	0.0352 (-79%)	0.0944 (-77%)	0.0470 (-78%)
	Median	0.2312 (-9%)	0.1545 (-10%)	0.3510 (-9%)	0.1924 (-10%)	0.1485 (-42%)	0.0943 (-45%)	0.2725 (-29%)	0.1339 (-37%)	0.0371 (-85%)	0.0233 (-87%)	0.0732 (-81%)	0.0346 (-84%)
	Norm	0.1972 (-17%)	0.1305 (-18%)	0.3181 (-18%)	0.1691 (-18%)	0.1410 (-41%)	0.0981 (-38%)	0.2153 (-45%)	0.1216 (-41%)	0.0530 (-78%)	0.0340 (-79%)	0.1018 (-74%)	0.0496 (-76%)
	TrimM	0.2269	0.1488	0.3489	0.1876	0.0647	0.0407	0.1198	0.0582	0.0361	0.0213	0.0721	0.0328
	Krum	0.1941 (+1%)	0.1235 (+3%)	0.3065 (0%)	0.1596 (+1%)	0.0255 (-87%)	0.0134 (-89%)	0.0456 (-85%)	0.0199 (-87%)	0.0509 (-73%)	0.0295 (-75%)	0.0891 (-71%)	0.0418 (-73%)
ML1M	Mean	0.1151 (-63%)	0.0702 (-66%)	0.2149 (-54%)	0.1022 (-60%)	0.0907 (-71%)	0.0549 (-73%)	0.1679 (-64%)	0.0793 (-69%)	0.0730 (-77%)	0.0437 (-79%)	0.1366 (-70%)	0.0640 (-75%)
	Median	0.2955 (-5%)	0.1975 (-4%)	0.4422 (-5%)	0.2446 (-4%)	0.0394 (-87%)	0.0228 (-89%)	0.0839 (-82%)	0.0370 (-85%)	0.0457 (-85%)	0.0270 (-87%)	0.0919 (-80%)	0.0418 (-84%)
	Norm	0.3000 (-2%)	0.1981 (-2%)	0.4465 (-2%)	0.2453 (-2%)	0.2901 (-5%)	0.1893 (-7%)	0.4306 (-5%)	0.2347 (-6%)	0.1442 (-53%)	0.0989 (-51%)	0.2104 (-54%)	0.1202 (-52%)
	TrimM	0.2593 (-17%)	0.1765 (-14%)	0.4151 (-10%)	0.2262 (-11%)	0.0391 (-87%)	0.0222 (-89%)	0.0838 (-82%)	0.0364 (-86%)	0.0445 (-86%)	0.0255 (-88%)	0.0863 (-81%)	0.0390 (-85%)
	Krum	0.2361 (0%)	0.1504 (0%)	0.3586 (-4%)	0.1899 (-3%)	0.0368 (-84%)	0.0216 (-86%)	0.0776 (-79%)	0.0346 (-82%)	0.0462 (-80%)	0.0268 (-82%)	0.0929 (-75%)	0.0418 (-79%)
Steam	Mean	0.0677 (-88%)	0.0403 (-89%)	0.1276 (-82%)	0.0596 (-86%)	0.0685 (-88%)	0.0408 (-89%)	0.1287 (-81%)	0.0601 (-86%)	0.069 (-88%)	0.0411 (-89%)	0.129 (-81%)	0.0603 (-86%)
	Median	0.1719 (-38%)	0.1205 (-38%)	0.2323 (-54%)	0.1400 (-47%)	0.0442 (-84%)	0.0265 (-86%)	0.0791 (-84%)	0.0376 (-86%)	0.0290 (-90%)	0.0175 (-91%)	0.0568 (-89%)	0.0262 (-90%)
	Norm	0.0717 (-87%)	0.0428 (-88%)	0.1322 (-81%)	0.0622 (-84%)	0.0690 (-87%)	0.0409 (-88%)	0.1292 (-81%)	0.0602 (-85%)	0.0682 (-87%)	0.0408 (-88%)	0.1300 (-81%)	0.0605 (-85%)
	TrimM	0.2001 (-65%)	0.1622 (-57%)	0.2502 (-64%)	0.1783 (-58%)	0.0378 (-93%)	0.0228 (-94%)	0.0714 (-90%)	0.0335 (-92%)	0.0288 (-95%)	0.0175 (-95%)	0.0584 (-92%)	0.0269 (-94%)
	Krum	0.1607 (-37%)	0.1253 (-29%)	0.2118 (-55%)	0.1416 (-42%)	0.0381 (-85%)	0.0237 (-87%)	0.0709 (-85%)	0.0341 (-86%)	0.0290 (-89%)	0.0175 (-90%)	0.0570 (-88%)	0.0264 (-89%)

Table 7: Recommendation performance under Spattack-O-S. We also report the performance drop rate w.r.t. clean model.

Dataset	Defense	1%				3%				5%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0647 (-74%)	0.0373 (-77%)	0.1421 (-65%)	0.0618 (-71%)	0.0594 (-76%)	0.0362 (-78%)	0.0997 (-75%)	0.0492 (-77%)	0.0541 (-78%)	0.0318 (-81%)	0.1039 (-74%)	0.0479 (-78%)
	Median	0.2365 (-7%)	0.1591 (-8%)	0.3606 (-6%)	0.1989 (-7%)	0.1994 (-22%)	0.1312 (-24%)	0.3075 (-20%)	0.1660 (-22%)	0.1198 (-53%)	0.0720 (-58%)	0.2068 (-46%)	0.0996 (-54%)
	Norm	0.2216 (-7%)	0.1417 (-11%)	0.3446 (-11%)	0.1811 (-13%)	0.1994 (-16%)	0.1249 (-22%)	0.3404 (-12%)	0.1698 (-18%)	0.1538 (-35%)	0.1032 (-35%)	0.2418 (-38%)	0.1314 (-37%)
ML1M	Mean	0.1204 (-61%)	0.0738 (-64%)	0.2230 (-52%)	0.1065 (-58%)	0.0925 (-70%)	0.0553 (-73%)	0.1753 (-62%)	0.0817 (-68%)	0.0805 (-74%)	0.0493 (-76%)	0.1568 (-66%)	0.0736 (-71%)
	Median	0.2995 (-4%)	0.2001 (-2%)	0.4452 (-4%)	0.2468 (-3%)	0.2439 (-22%)	0.1570 (-23%)	0.3641 (-21%)	0.1957 (-23%)	0.0447 (-86%)	0.0264 (-87%)	0.0897 (-81%)	0.0407 (-84%)
	Norm	0.3028 (-1%)	0.1986 (-2%)	0.4520 (-1%)	0.2468 (-2%)	0.2902 (-5%)	0.1898 (-6%)	0.4382 (-4%)	0.2375 (-5%)	0.2023 (-34%)	0.1402 (-31%)	0.2793 (-39%)	0.1648 (-34%)
Steam	Mean	0.0701 (-88%)	0.0410 (-89%)	0.1404 (-80%)	0.0635 (-85%)	0.0671 (-88%)	0.0390 (-90%)	0.1308 (-81%)	0.0593 (-86%)	0.0695 (-88%)	0.0410 (-89%)	0.1348 (-81%)	0.0617 (-85%)
	Median	0.3333 (+20%)	0.2448 (+27%)	0.4602 (-9%)	0.2855 (+8%)	0.0266 (-90%)	0.0152 (-92%)	0.0600 (-88%)	0.0258 (-90%)	0.0218 (-92%)	0.0115 (-94%)	0.0498 (-90%)	0.0204 (-92%)
	Norm	0.1761 (-67%)	0.1226 (-64%)	0.2673 (-61%)	0.1520 (-61%)	0.0685 (-87%)	0.0398 (-88%)	0.1324 (-81%)	0.0602 (-85%)	0.0703 (-87%)	0.0414 (-88%)	0.1359 (-80%)	0.0623 (-84%)

the degradation will further increase to 82% when  $\rho = 5\%$ . The reason is that these tailed items in FR have lower defense breaking points and can be easily broken.

### Performance Evaluation of Spattack-O-S

The detailed results of Spattack-O-S are reported in Tab. 7, where each malicious client only uploads malicious gradients for partial items to avoid triggering the anomaly detection based on the user’s degree. We restrict the  $\tilde{m}_{max}$  as the maximum number of uploading items in benign clients. As seen, Spattack-O-S can still significantly degrade recommendation performance under 1% malicious clients and prevent the model convergence for a 5% ratio. For example, Spattack-O-S achieves 35% (ML100K), 31% (ML1M), and 80% (Steam) drop rate at least when the malicious client ratio  $\rho$  is 5%. Overall, Spattack-O-S can bypass anomaly

detection based on the user’s degree and still achieve successful attacks with few malicious clients.

### Performance Evaluation of Spattack-L-D

Considering non-omniscient attackers, where the benign gradients are unavailable, we launch the Spattack-L-D by randomly generating Gaussian noise as malicious gradients. To get comparable results, the ratios of malicious clients in Spattack-L-D are set to higher values of {5%, 10%, 15%}. As seen in Tab. 8, though equipped with statically robust aggregators, Spattack-L-D can still prevent the convergence with degradation of 81% to 97% under a 10% ratio. The results also provide a valuable implication that hiding the gradients of benign clients cannot protect the federated recommender well, because the attackers can break down the model by using random noise as a substitution. Besides, one

Table 8: Recommendation performance under Spattack-L-D. We also report the performance drop ratio w.r.t. clean model.

Dataset	Defense	5%				10%				15%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0318 (-87%)	0.0160 (-90%)	0.0785 (-81%)	0.0309 (-85%)	0.0233 (-91%)	0.0121 (-93%)	0.0562 (-86%)	0.0225 (-89%)	0.0265 (-89%)	0.0139 (-92%)	0.0562 (-86%)	0.0231 (-89%)
	Median	0.2163 (-15%)	0.1324 (-23%)	0.3637 (-6%)	0.1797 (-16%)	0.0095 (-96%)	0.0056 (-97%)	0.0233 (-94%)	0.0099 (-95%)	0.0233 (-91%)	0.0139 (-92%)	0.0530 (-86%)	0.0234 (-89%)
	Norm	0.2418 (+2%)	0.1480 (-7%)	0.3659 (-6%)	0.1876 (-10%)	0.1516 (-36%)	0.0910 (-43%)	0.2969 (-23%)	0.138 (-33%)	0.106 (-55%)	0.0601 (-62%)	0.1877 (-52%)	0.086 (-59%)
	TrimM	0.2269 (-10%)	0.1511 (-8%)	0.3690 (-9%)	0.1968 (-8%)	0.0074 (-97%)	0.0041 (-97%)	0.0286 (-93%)	0.0110 (-95%)	0.0276 (-89%)	0.0162 (-93%)	0.0551 (-86%)	0.0249 (-88%)
	Krum	0.1474 (-23%)	0.0879 (-27%)	0.2259 (-27%)	0.1132 (-28%)	0.0159 (-92%)	0.0101 (-92%)	0.0318 (-90%)	0.0152 (-90%)	0.0286 (-85%)	0.0159 (-87%)	0.0615 (-80%)	0.0264 (-83%)
ML1M	Mean	0.0272 (-91%)	0.0153 (-93%)	0.0720 (-84%)	0.0295 (-88%)	0.0237 (-92%)	0.0128 (-94%)	0.0523 (-89%)	0.0219 (-91%)	0.0260 (-92%)	0.0146 (-93%)	0.0603 (-87%)	0.0254 (-90%)
	Median	0.0121 (-96%)	0.0055 (-97%)	0.0894 (-81%)	0.0296 (-88%)	0.024 (-92%)	0.0144 (-93%)	0.046 (-90%)	0.0215 (-92%)	0.0387 (-88%)	0.0220 (-89%)	0.0732 (-84%)	0.0330 (-87%)
	Norm	0.2805 (-8%)	0.1822 (-10%)	0.4333 (-5%)	0.2313 (-8%)	0.1962 (-36%)	0.1185 (-42%)	0.3343 (-27%)	0.1628 (-35%)	0.1119 (-63%)	0.0664 (-67%)	0.2240 (-51%)	0.1023 (-59%)
	TrimM	0.0232 (-93%)	0.0103 (-95%)	0.1611 (-65%)	0.0537 (-79%)	0.0359 (-88%)	0.0207 (-90%)	0.0717 (-85%)	0.0321 (-87%)	0.0255 (-92%)	0.0152 (-93%)	0.0503 (-89%)	0.0231 (-91%)
Steam	Mean	0.0205 (-96%)	0.0114 (-97%)	0.0384 (-94%)	0.0170 (-96%)	0.0226 (-96%)	0.0129 (-97%)	0.0520 (-93%)	0.0221 (-95%)	0.0274 (-95%)	0.0155 (-96%)	0.0592 (-91%)	0.0256 (-94%)
	Median	0.0285 (-90%)	0.0160 (-92%)	0.0549 (-89%)	0.0245 (-91%)	0.0434 (-84%)	0.0261 (-87%)	0.0874 (-83%)	0.0400 (-85%)	0.0493 (-82%)	0.0293 (-85%)	0.0906 (-82%)	0.0426 (-84%)
	Norm	0.0171 (-97%)	0.0098 (-97%)	0.0442 (-94%)	0.0184 (-95%)	0.0226 (-96%)	0.0121 (-96%)	0.0560 (-92%)	0.0227 (-94%)	0.0250 (-95%)	0.0134 (-96%)	0.0634 (-91%)	0.0256 (-94%)
	TrimM	0.0296 (-95%)	0.0166 (-96%)	0.0552 (-92%)	0.0248 (-94%)	0.0440 (-92%)	0.0263 (-93%)	0.0879 (-87%)	0.0402 (-90%)	0.0480 (-92%)	0.0285 (-93%)	0.0903 (-87%)	0.0421 (-90%)
	Krum	0.0288 (-89%)	0.0168 (-90%)	0.0554 (-88%)	0.0253 (-90%)	0.0434 (-83%)	0.0265 (-85%)	0.0866 (-81%)	0.0402 (-84%)	0.0472 (-81%)	0.0281 (-84%)	0.0895 (-81%)	0.0416 (-83%)

Table 9: Recommendation performance under Spattack-L-S. We also report the performance drop rate w.r.t. clean model.

Dataset	Defense	5%				10%				15%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0742 (-70%)	0.0425 (-74%)	0.1559 (-62%)	0.0683 (-68%)	0.0456 (-82%)	0.0249 (-85%)	0.1124 (-72%)	0.0462 (-78%)	0.0477 (-81%)	0.0264 (-84%)	0.0997 (-75%)	0.0428 (-80%)
	Median	0.2238 (-12%)	0.1433 (-17%)	0.3733 (-3%)	0.1913 (-11%)	0.0308 (-88%)	0.0179 (-90%)	0.0764 (-80%)	0.0321 (-85%)	0.0339 (-87%)	0.0206 (-88%)	0.0742 (-81%)	0.0336 (-84%)
	Norm	0.2397 (+1%)	0.1525 (-4%)	0.3690 (-5%)	0.1935 (-7%)	0.2068 (-13%)	0.1243 (-22%)	0.3118 (-20%)	0.1579 (-24%)	0.1315 (-45%)	0.0804 (-50%)	0.2503 (-36%)	0.1187 (-43%)
ML1M	Mean	0.0359 (-88%)	0.0207 (-90%)	0.0917 (-80%)	0.0385 (-85%)	0.0268 (-91%)	0.0156 (-92%)	0.0641 (-86%)	0.0274 (-89%)	0.0288 (-91%)	0.0166 (-92%)	0.0679 (-85%)	0.0290 (-89%)
	Median	0.0825 (-74%)	0.039 (-81%)	0.2237 (-52%)	0.0840 (-67%)	0.0285 (-91%)	0.0169 (-92%)	0.0536 (-88%)	0.0250 (-90%)	0.0387 (-88%)	0.0229 (-89%)	0.0728 (-84%)	0.0338 (-87%)
	Norm	0.2815 (-8%)	0.1822 (-10%)	0.4368 (-4%)	0.2321 (-7%)	0.2063 (-33%)	0.1246 (-38%)	0.3464 (-24%)	0.1696 (-32%)	0.1281 (-58%)	0.0746 (-63%)	0.2445 (-46%)	0.1118 (-55%)
Steam	Mean	0.0453 (-92%)	0.0259 (-93%)	0.0901 (-87%)	0.0401 (-90%)	0.0440 (-92%)	0.0258 (-93%)	0.0914 (-87%)	0.0408 (-90%)	0.0410 (-93%)	0.0239 (-94%)	0.0890 (-87%)	0.0393 (-91%)
	Median	0.0378 (-86%)	0.0221 (-89%)	0.0791 (-84%)	0.0351 (-87%)	0.0488 (-82%)	0.0285 (-85%)	0.0927 (-82%)	0.0426 (-84%)	0.0450 (-84%)	0.0264 (-86%)	0.0938 (-81%)	0.0420 (-84%)
	Norm	0.0679 (-87%)	0.0389 (-89%)	0.1628 (-76%)	0.0692 (-82%)	0.0554 (-90%)	0.0322 (-91%)	0.1138 (-84%)	0.0508 (-87%)	0.0469 (-91%)	0.0265 (-92%)	0.1031 (-85%)	0.0444 (-89%)

can observe that Norm aggregators can provide better defense than others. The reason is that the malicious gradients of Spattack-L-D are from a Gaussian noise with the same variance: a large variance will benefit skewing data under statically robust aggregators but can be easily clipped by the norm-based defense.

### Performance Evaluation of Spattack-L-S

When both knowledge and capability are limited, Spattack-L-S still significantly degrades the FR model as shown in Tab. 9. The performance drops by about 56%, 73% and 78% under 5%, 10%, 15% malicious ratio on average, indicating that the FR system is vulnerable to our attacks, which could hinder its applicability in various domains.

### More Results on the Transferability of Attacks

Here, we evaluate the effectiveness of Spattack on more FR scenarios. First, we perform Spattack with a malicious ra-

tio of 10% to the SOTA FedGNN (Wu et al. 2021), where the Byzantine clients can only upload malicious gradients of item embeddings. No Defense and Defense correspond to mean and median aggregators, respectively. As shown in Fig. 6, we find that FedGNN’s performance dramatically drops under Spattack, demonstrating the common vulnerability of FedMF and FedGNN. Even though GNN’s parameters are densely aggregated, attackers can still prevent model convergence by only poisoning item embeddings. Moreover, we also show the effectiveness of Spattack under the Adam optimizer in Fig. 9. Lastly, we evaluate Spattack’s effectiveness when differential privacy is applied to the local gradients in (Wu et al. 2021). As shown in Fig. 8, Spattack still achieves successful attacks.

### More results on the Hyperparameter Analysis

In Fig. 7, we present the convergence of FR under defense (i.e., Median AGR) against Spattack with a malicious ratio

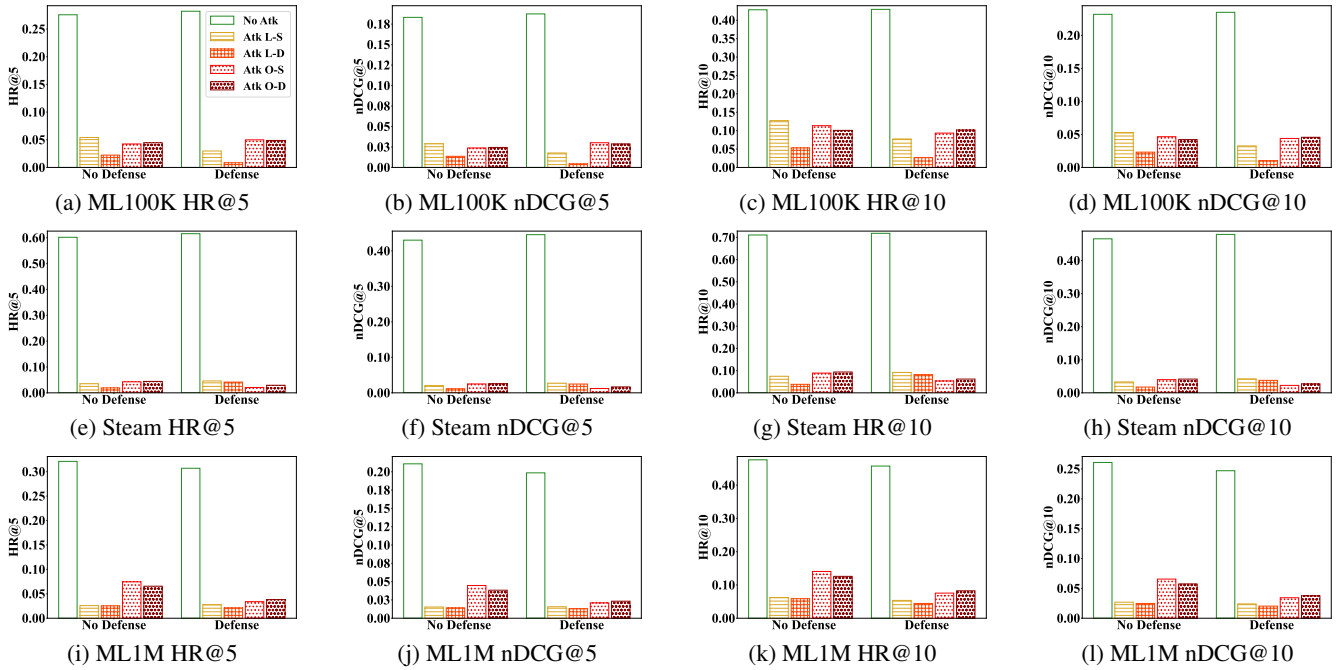


Figure 6: Experimental Results on FedGNN Model.

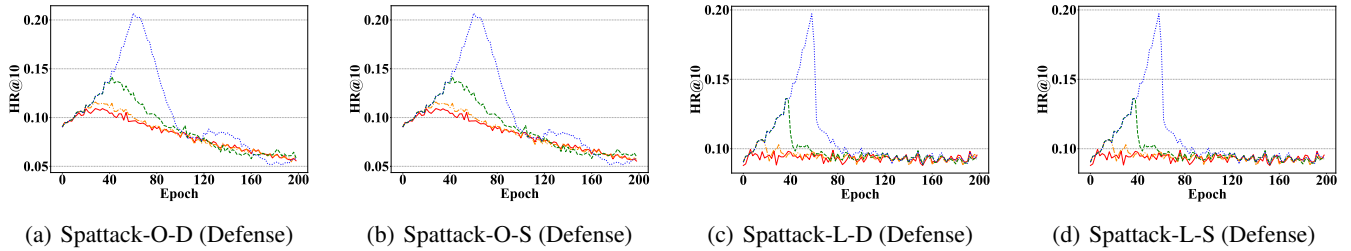


Figure 7: Performance of Spattack starting with different epochs on Steam dataset.

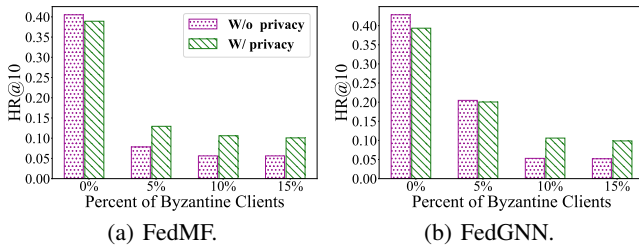


Figure 8: Attack performance under differential privacy.

of 10%. The results correspond to starting attacks at 0, 20, 40, and 60, with a visualization of HR@10 in 200 epochs on Steam. We observe that the Median AGR directly picks the malicious gradient as output and rapidly decreases the model performance along the opposite direction of the true gradient, which once again validates the vulnerability of sparse aggregation on FR.

### The effectiveness under lower malicious ratio

To explore the lowest malicious ratio for the effectiveness of Spattack, we consider the worst-case scenario, Spattack-O, where the attacker have the knowledge of total gradients.

Table 10: The performance under lower malicious ratios.

AGR(-)	Spattack	Clean	1 (0.1%)	5 (0.5%)	9 (1%)	29 (3%)	49 (5%)
Mean	O-D	0.4051	0.0997 (-75.4%)	0.1029 (-74.6%)	0.0986 (-75.7%)	0.0944 (-76.7%)	0.0944 (-76.7%)
	O-S	0.4051	0.2471 (-39.0%)	0.175 (-56.8%)	0.1421 (-64.9%)	0.0997 (-75.4%)	0.1039 (-74.4%)
Median	O-D	0.3849	0.3818 (-0.8%)	0.3743 (-2.8%)	0.3510 (-8.8%)	0.2725 (-29.2%)	0.0732 (-81.0%)
	O-S	0.3849	0.3826 (-0.6%)	0.3765 (-2.2%)	0.3606 (-6.3%)	0.3075 (-20.1%)	0.2068 (-46.3%)

We report HR@10 on the ML-100K dataset under varying malicious ratios: 0.1%, 0.5%, 1%, 3%, and 5%, corresponding to 1, 5, 9, 29, and 49 malicious clients, respectively. The experimental results are reported in Tab. 10. We have the following observations:

- Without defense (i.e., Mean), with only 0.1% malicious ratio (1 malicious client), Spattack-O-D dramatically degrades the performance by over 75% and prevents model convergence, which is consistent with Proposition 1. Even if the capability is limited, Spattack-O-S still effectively degrades the performance by 39% with only 1 malicious client.
- For the defense aggregator (i.e., Median), a 3% malicious ratio (29 malicious clients) can significantly degrade model performance by over 20%.

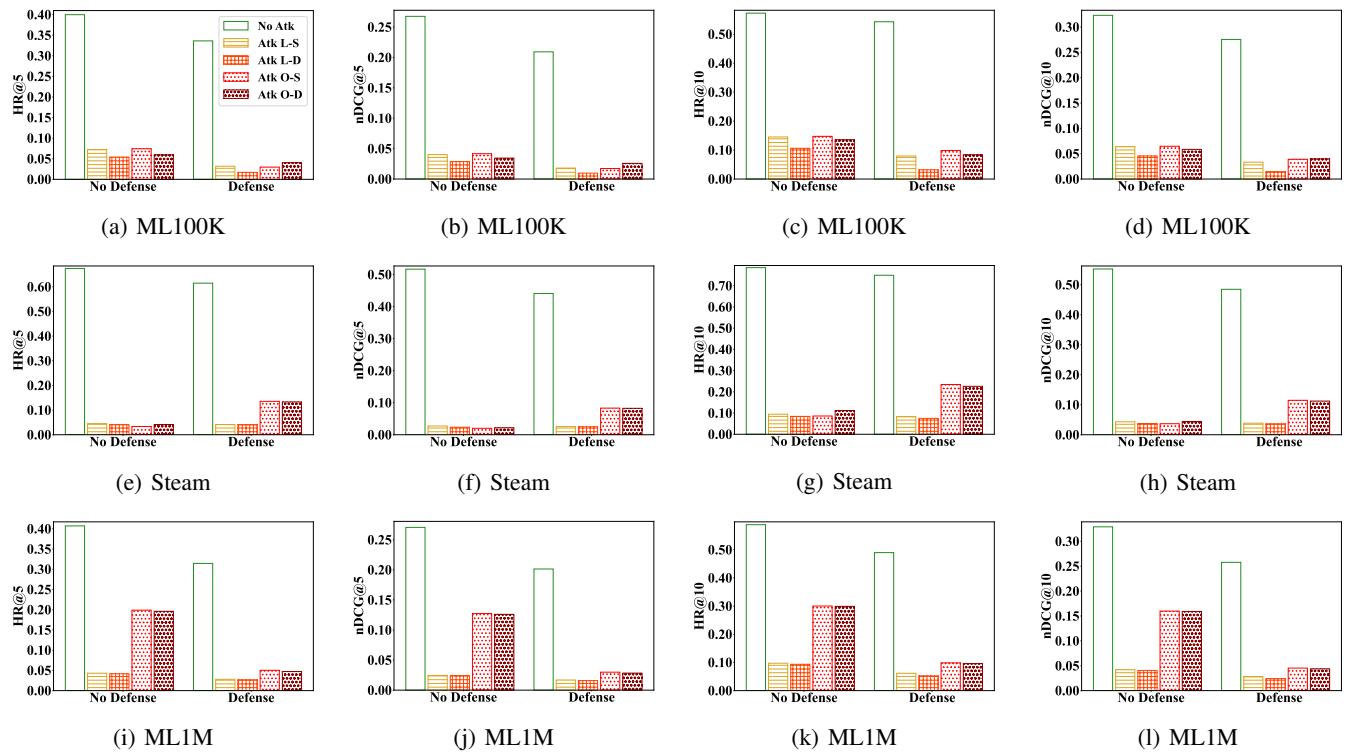


Figure 9: Results on FR Model with Adam Optimizer.