Artifact Evaluated
NDSS
Available
Functional
Reproduced

RAIFLE: Reconstruction Attacks on Interaction-based Federated Learning with Adversarial Data Manipulation

Dzung Pham, Shreyas Kulkarni, Amir Houmansadr
University of Massachusetts Amherst
{dzungpham, svkulkarni, amir}@cs.umass.edu

***Abstract*—Federated learning has emerged as a promising privacy-preserving solution for machine learning domains that rely on *user interactions*, particularly recommender systems and online learning to rank. While there has been substantial research on the privacy of traditional federated learning, little attention has been paid to the privacy properties of these interaction-based settings. In this work, we show that users face an elevated risk of having their private interactions reconstructed by the central server when the server can control the training features of the items that users interact with. We introduce RAIFLE, a novel optimization-based attack framework where the server actively manipulates the features of the items presented to users to increase the success rate of reconstruction. Our experiments with federated recommendation and online learning-to-rank scenarios demonstrate that RAIFLE is significantly more powerful than existing reconstruction attacks like gradient inversion, achieving high performance consistently in most settings. We discuss the pros and cons of several possible countermeasures to defend against RAIFLE in the context of interaction-based federated learning. Our code is open-sourced at `https://github.com/dzungvpham/raifle`.**

I. Introduction

Federated learning (FL) [46], [35] is an emerging approach to building privacy-preserving recommender systems (RS) [81], [68] and online learning to rank (OLTR) solutions [39], [74]. In such systems, users interact with server-prepared "items" (e.g., news articles, media, and products) via clicks, ratings, and other types of interactions, then train their FL model using these private interactions. As a result, users can benefit from a better item ranking/recommendation experience without having to share potentially sensitive interaction data with service providers. We broadly define this variation on FL as *interaction-based FL* (IFL). Unlike traditional FL, users in IFL do not inherently own any data other than their interactions with the server-controlled items. As the two most prominent IFL instances, RS and OLTR have many far-reaching applications such as web search (e.g., Google), online advertising (e.g., Facebook ads), and e-commerce (e.g., Amazon), which often rely on vast amounts of private and sensitive user data. Consequently, the use of IFL for RS/OLTR has gained popularity in both academia and industry, with a wide variety of algorithms proposed such as federated collaborative filtering [4], [55], federated graph neural net for recommendation [80], reinforcement learning-based federated OLTR [39], etc. Their potential to leverage data from diverse sources while maintaining user privacy and data ownership makes them a compelling approach to enhancing recommendation/ranking accuracy and addressing data silo challenges.

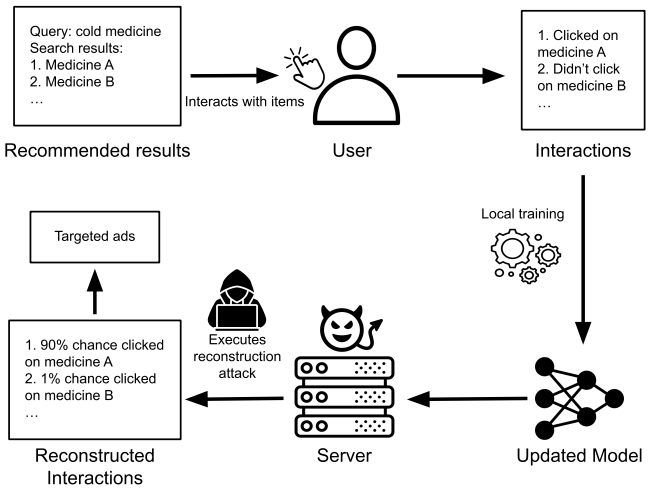

Fig. 1. Example of an interaction reconstruction attack in federated recommendation/learning-to-rank. A malicious server may infer user interactions from the FL updates to execute targeted advertising.

IFL's departure from standard FL thus poses a significant privacy risk that is currently not well understood. Previous work on federated RS/OLTR [68], [82], [74] has often overlooked the server's knowledge and control over the items presented to users. This unique aspect of IFL is also not considered in existing FL privacy attacks like gradient inversion [33], [84], [52], since in traditional FL, the server usually cannot exert a great deal of influence on users' data or behaviors. We argue that the IFL server should not require access to user interactions for two key reasons. Firstly, the server does not always need user interaction data to provide essential services. For example, web search engine providers do not need to know precisely which links users click on once they have delivered the search results to the users. Secondly, the server can either accidentally or intentionally leak private user interactions and preferences, resulting in serious consequences such as compro-

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240363
www.ndss-symposium.org

mising user anonymity [48], exposing sensitive interests [79], and enabling targeted advertising [72], ultimately leading to the loss of user trust (Figure 1). Addressing these privacy challenges is crucial to ensure the responsible and secure implementation of FL for RS and OLTR.

In this work, we aim to study the risk of reconstructing user interaction data in IFL. We present **RAIFLE**,[1] a general **r**econstruction **a**ttack for **i**nteraction-based **f**ederated **le**arning systems. Similar to the gradient inversion attacks in traditional FL [33], [84], [52], RAIFLE aims to find the most likely interactions by minimizing the "distance" between the simulated local update created with candidate interactions and the actual received local update. However, unlike gradient inversion, RAIFLE exploits the IFL server's knowledge and control over the items to execute *Adversarial Data Manipulation* (ADM), a novel attack vector where the server modifies the training data features associated with the items to produce adversarial behaviors in the local updates. We design two ADM techniques: the *fingerprint* method where the IFL server selectively chooses which training features to keep or "zero out" to control the local gradients and the *noise injection* method where the training features are replaced with random noise. We empirically show that this active adversarial attack can outperform vanilla gradient inversion as well as undermine existing privacy-enhancing mechanisms such as secure aggregation [14] and private information retrieval (PIR) [19], [40]. Our attack can also work even when the server does not have direct control over the training features, particularly when the users extract the features themselves. To the best of our knowledge, RAIFLE is the first optimization-based active reconstruction attack applicable to not only federated RS/OLTR but also the more general IFL setting.

In light of these findings, we discuss the pros and cons of various potential countermeasures to alleviate the risks of our privacy attacks, including local differential privacy, secure aggregation, data validation, personalization, etc. We hope our research can inform future endeavors to develop more secure and private federated RS/OLTR systems specifically and IFL in general. To summarize, our main contributions are:

- We identify the server's knowledge and control over the data items in interaction-based FL (IFL) scenarios like federated RS/OLTR as a privacy vulnerability that can facilitate stronger reconstruction attacks.
- We introduce RAIFLE, a general optimization-based reconstruction attack framework for IFL. RAIFLE utilizes Adversarial Data Manipulation (ADM), a novel attack vector unique to IFL where the IFL server actively manipulates the items to increase RAIFLE's attack performance.
- We evaluate RAIFLE in two representative IFL systems, namely federated RS and OLTR, and show that our attack has strong inference performance under most tested scenarios, even when the server can only indirectly influence the training features.
- We analyze various countermeasures to mitigate privacy leakage against RAIFLE and ADM for federated RS/OLTR and IFL in general.

[1]Pronounced like "rifle"

II. Background and Related Work

We provide a brief overview of privacy attacks and defenses in FL and a description of IFL.

A. Attacks on FL

1) Passive Attacks: Federated Learning (FL) is a decentralized approach to machine learning that allows each participant to collaboratively train a machine learning model without having to share their private data with a central server [46], [35]. Despite the decentralization of data, research has shown that FL can be vulnerable to a class of attack called **gradient inversion** [33], [84], [52], which allows an honest-but-curious server to invert users' gradients to find an approximation of users' local data by solving an optimization problem of the following form:

$$\underset{\mathcal{X}'}{\operatorname{argmin}} \left[\mathbf{dist}(\nabla_\theta^{\mathcal{X}'}, \nabla_\theta^{\mathcal{X}}) + \rho(\mathcal{X}') \right] \tag{1}$$

where $\mathcal{X}'$ is the server's approximation of the user's data, $\mathcal{X}$ is the user's actual data, θ represents the model parameters, $\nabla_\theta^{\mathcal{X}'}$ and $\nabla_\theta^{\mathcal{X}}$ are the gradients w.r.t. θ when trained on $\mathcal{X}'$ and $\mathcal{X}$ (respectively), **dist** is a measure of distance between the gradients, and ρ is the prior/regularizer placed on $\mathcal{X}'$. Essentially, the server tries to find an approximation $\mathcal{X}'$ of the local data $\mathcal{X}$ that would lead to the simulated gradients $\nabla_\theta^{\mathcal{X}'}$ closest to the actual received gradients $\nabla_\theta^{\mathcal{X}}$. Hence, the attack is also called "gradient matching".

The majority of research in gradient inversion has so far been focused on deep neural networks for computer vision and natural language processing tasks [52], which differ from RS and OLTR in the following ways: (a) user interactions in RS and OLTR research are typically structured discrete data (e.g., clicks vs no clicks), unlike images and texts; (b) RS/OLTR models can have much fewer parameters and are also not restricted to neural nets. These differences can render existing gradient inversion attacks not immediately applicable to the RS/OLTR context.

2) Active Attacks: Unlike gradient inversion, which only passively observes the local updates without changing any aspects of the FL protocol, active FL attacks involve a malicious server that deliberately modifies parts of the FL training process to further enable user data reconstruction. Most notable is **model manipulation** attacks [22], [53], [13], in which the server directly manipulates the FL model's architecture or weights in such a way that the user data is more easily leaked through the FL updates. This class of attack has a much higher reconstruction quality than pure gradient inversion and can even be applied to FL with secure aggregation defenses. Another type of active attack is Sybil-based attacks, where the server introduces fake FL participants that are completely under the server's control [12]. By choosing only one real user and setting the FL updates from all other fake users to 0, the server can easily isolate the real user's local updates, thus bypassing both secure aggregation and differential privacy.

3) Other Attacks: Deep neural networks (DNN) for computer vision tasks have been demonstrated to be vulnerable to **adversarial perturbations** in their image inputs, which can cause the models to diverge from expected behaviors [83], [2], [3], [38]. An image's deep representations extracted from

a DNN can in fact be manipulated via this attack to closely resemble those of any arbitrary image [65]. DNNs for natural language processing tasks are also susceptible to adversarial inputs, although the attack techniques are rather different from image-based attacks due to the discrete nature of the input space [85], [77]. While adversarial perturbation attacks are not commonly applied in traditional FL, it is particularly applicable to our data manipulation techniques for IFL given the server's control of the interaction items. Another attack closely related to our problem of reconstructing user interactions is **membership inference attacks** (MIA) against ML models [67], [49], which aims to determine whether a record is part of the training data.

B. Privacy Defenses for FL

1) Differential Privacy: Considered the state-of-the-art privacy model and defense in statistics and machine learning, differential privacy (DP) [21], [1] has been applied in numerous FL applications [47], [58], [35]. Formally, a randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Z}$ is (ε, δ)-differentially private if for any pair of database $x, x' \in \mathcal{X}$ differing in *at most* one record and for any $S \subseteq \mathcal{Z}$, we have:

$$\mathbb{P}[\mathcal{M}(x) \in S] \leq e^{\varepsilon}\mathbb{P}[\mathcal{M}(x') \in S] + \delta \tag{2}$$

where $\varepsilon \geq 0$ represents the privacy budget and $\delta \in [0, 1)$ represents the probability of privacy leakage. In FL, the randomized mechanism $\mathcal{M}$ is typically the training process that occurs on users' devices and the aggregation process.

The definition above describes the *central* DP model, which in the context of FL means that users trust a third-party curator with the collection, aggregation, and privatization of their local updates before releasing them to the server. In this paper, we rely on the stricter but more realistic **local differential privacy** model (LDP), where users do not trust any entity with their data [37]. Formally, LDP requires equation 2 to hold for *any* pair of x, x'. While LDP can effectively defend against gradient inversion attacks [84], it often incurs a high utility cost [78]. Practical levels of ε for LDP therefore tend to be "in the hundreds" [11], [28] in order to achieve a decent model utility. Achieving a good balance between utility and privacy thus remains a challenging problem [20].

2) Secure Aggregation: Another line of privacy defense for FL is secure aggregation (SA) [14], [10], which relies on cryptography techniques to securely aggregate (e.g., summing) the local updates before sharing them with the server. With this mechanism, the server can only observe the aggregated result without knowing the individual contributions from any user. SA, therefore, aims to achieve two objectives: "privacy by aggregation" – to make it difficult for the server to infer useful information about individuals from the aggregated updates – and "privacy by shuffling" – to hide the link between individuals and updates so that even if any information can be inferred, it cannot be traced back to any particular user [53]. Nevertheless, research has shown that vanilla SA for FL is susceptible to model manipulation attacks [22], [53], [13], as the final aggregated model can be adversarially engineered to capture private local data without being affected by the aggregation. To prevent such attacks with more theoretical guarantees, SA needs to be combined with user-applied noise to achieve distributed differential privacy [34], which requires less noise than pure LDP and thus better preserves FL utility.

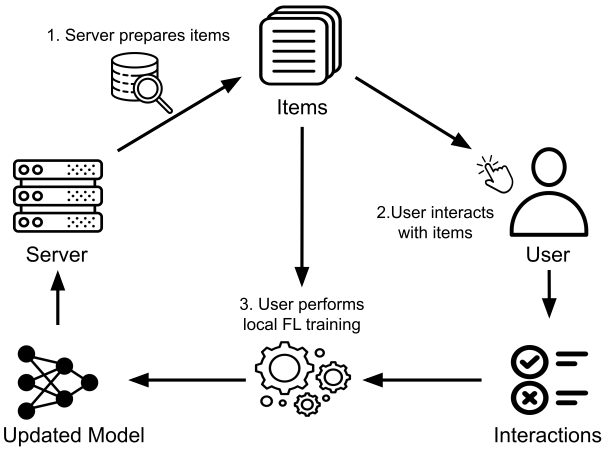

Fig. 2. Diagram of Interaction-based Federated Learning (IFL). Users interact with server-prepared items and train the FL model using the items and their private interactions. Users may apply privacy defense techniques such as differential privacy before sending local updates to the server.

C. Interaction-based FL (IFL)

We broadly define interaction-based FL (IFL) as a variation of FL in which users *interact* with items provided by the server and perform the local training using their interaction information (Figure 2). The nature of the items depends on the specific application (e.g., news articles for news recommendations or search results for web search). Unlike traditional FL, IFL has two distinctive characteristics: (1) the server can influence how the items are (re)presented to the users, and (2) the primary data to be protected is the user interactions. We describe below two major instances of IFL: federated recommendation systems and federated online learning to rank.

1) Federated Recommender Systems: Recommender systems (RS) is a technology that assists users with discovering relevant items like products, news, media, etc. [63]. RS typically relies on user-item interactions, which unfortunately have been shown to be capable of revealing sensitive user attributes (e.g., age, gender) [79] and even uniquely identifying users [48]. To keep user interactions protected, recent research has looked to FL as the foundation to build more privacy-preserving recommendation services [81], [68].

One of the earliest and most influential RS algorithms is **collaborative filtering** (CF), which recommends items based on the behaviors and preferences of similar users [24]. A common approach to CF is *matrix factorization* [17]: essentially, the user-item interaction matrix is factored into two lower-dimensional latent factor matrices, one for the user embeddings and one for the item embeddings. Building upon CF, **federated collaborative filtering** (FCF) formulates the item matrix as public parameters (i.e., known by all participants) and the user matrix as private parameters, which are privately updated by each user without being shared with anyone else [4]. This idea has been incorporated in various subsequent works such as federated neural collaborative filtering (FNCF) [55] and federated graph neural net for recommendation (FedGNN) [80]. However, keeping user embeddings private by itself does not

automatically guarantee privacy for users since their interactions can still be inferred as demonstrated by previous research [18], [82] (as well as our own work).

2) Federated Online Learning to Rank: Learning to rank (LTR) is a machine learning task whose aim is to learn ranking models for information retrieval systems and has been applied to a variety of applications such as recommendation and web search [45]. There are three major LTR approaches: (a) the *pointwise* approach compares each item to a relevance label, (b) *pairwise* compares pairs of items with different preferences, and (c) *listwise* considers the entire ranked list as a unit to optimize the overall ranking. In this paper, we focus on federated online learning to rank (FOLTR), which primarily learns from user interactions (i.e., implicit feedback), unlike traditional LTR which relies on labeled (i.e., explicit) relevance judgement [26], [39], [74]. Of particular interest is the **Federated Pairwise Differentiable Gradient Descent** (FPDGD) method, which proposes using the Pairwise Differentiable Gradient Descent (PDGD) algorithm [74], [51] in the FL setting. The authors of FPDGD claim a higher ranking performance than the existing state-of-the-art FOLTR algorithm [39], even with DP applied. (We note, however, that FPDGD's evaluation with DP is not fully substantiated as it contains two issues: (a) the Laplace mechanism used only satisfies central DP, whereas the mechanism used in [39] satisfies LDP, and (b) the authors clipped the L_2-norm of the model weights instead of L_1-norm as required by the Laplace mechanism.)

3) Attacks on IFL: To the best of our knowledge, only one paper has attempted to devise a privacy attack to steal user interactions in federated RS [82]. However, their proposed Interaction Membership Inference Attack (IMIA) is a heuristic-based search that does not make use of the gradients with respect to the simulated interactions, resulting in poor inference performance. Furthermore, the attack is limited to only federated RS and cannot be directly applied to other IFL scenarios such as OLTR.

III. Overview of RAIFLE

In this section, we describe the high-level approach of RAIFLE, our reconstruction attack on IFL. Descriptions of useful notations used throughout the subsequent sections are provided in Table I.

A. Threat Model

Our adversary is a FL server that not only knows which items are presented to which users but can also modify the representations of the items served to users, particularly the training features used in the local FL training. This threat model is consistent with previous research that involves active and malicious FL servers [14], [10], [53]. The server's knowledge of user-item impressions is realistic in most practical IFL systems like federated RS and OLTR. While it is technically possible to hide the user-item mapping via Private Information Retrieval (PIR) [19], the prohibitively expensive computational costs prevent such a technique from being widely adopted at scale. The ability to modify items is also justifiable for various business/operational reasons, such as the need to experiment with new/updated item features to improve service quality.

TABLE I. Notations

Symbol	Description
m	Number of items
n	Number of users
d	Dimension of item features or representations
p	Number of model parameters, assumed to be $\geq d$
$\mathcal{X}$	A matrix in $\mathbb{R}^{m\times d}$ that represents training features (or embeddings for RS) of the items
$\hat{\mathcal{X}}$	The user's updated item embeddings for RS
$\mathcal{I}$	A vector in $\mathbb{R}^m$ for the user's true interactions
$\mathcal{I}'$	The server's reconstructed interactions
θ	A vector in $\mathbb{R}^p$ for the global model parameters
$\hat{\theta}$	The user's updated model parameters
f	A function that represents the global model. Takes $\mathcal{X}$, θ, $\mathcal{I}$. Returns the output of the model.
g	The local FL algorithm. Takes $\mathcal{X}, \theta, \mathcal{I}$ (and user embedding in RS). Returns the updated model parameters (and the updated item embeddings for RS).
$\mathcal{L}_{atk}$	Loss function used by our attack
$\nabla_{\mathcal{I}'} g$	A $m \times p$ matrix for the gradients of g w.r.t. $\mathcal{I}'$
$\nabla^2_{\mathcal{I}'} g$	A third-order tensor for the Hessian of g w.r.t. $\mathcal{I}'$
$\nabla_{\mathcal{I}'} \mathcal{L}_{atk}$	$m \times 1$ vector for the gradients of $\mathcal{L}_{atk}$ w.r.t. $\mathcal{I}'$
$\nabla^2_{\mathcal{I}'} \mathcal{L}_{atk}$	$m \times m$ matrix for the Hessian of $\mathcal{L}_{atk}$ w.r.t. $\mathcal{I}'$

B. Basic Reconstruction Framework

We first present a high-level approach to RAIFLE without any server-side manipulation from the Bayesian perspective. Given a model f with global parameters θ, a user u, the items $\mathcal{X}$ presented to u, and u's updated model parameters $\hat{\theta}$ learned via a learning algorithm g, we are interested in inferring u's true interactions $\mathcal{I}$. Probabilistically speaking, we want to find:

$$\arg\max_{\mathcal{I}'} Pr(\mathcal{I}' \mid \mathcal{X}, \theta, \hat{\theta}) \tag{3}$$

Applying Bayes' theorem, we get:

$$Pr(\mathcal{I}' \mid \mathcal{X}, \theta, \hat{\theta}) \propto Pr(\hat{\theta} \mid \mathcal{X}, \theta, \mathcal{I}') \cdot Pr(\mathcal{I}' \mid \mathcal{X}, \theta) \tag{4}$$

Thus, we can approach the problem as a maximum a priori (MAP) estimation task, where we try to maximize the right-hand side (RHS) of equation 4. $\hat{\theta}$ can be considered the "data", $\mathcal{I}'$ can be considered the "parameters", and the second term of the RHS can be considered the prior on $\mathcal{I}'$. In our experiments, we use an uninformative (e.g., uniform) prior and focus on maximizing the first term of the RHS, i.e., the likelihood of $\hat{\theta}$ given $\mathcal{X}, \theta, \mathcal{I}'$ via optimization:

$$\arg\min_{\mathcal{I}'} \mathcal{L}_{atk}(\hat{\theta}, g(\mathcal{X}, \theta, \mathcal{I}')) \tag{5}$$

where $\mathcal{L}_{atk}$ is a loss function that measures the "distance" (e.g., L_2) between the user's learned parameters $\hat{\theta}$ and the server's simulated parameters.

If the interaction values are discrete (e.g., clicked vs. no clicked, integer rating), one approach is to brute-force through all possible $\mathcal{I}'$ to find the one that would produce the closest simulated parameters to $\hat{\theta}$, but this is not computationally feasible when $|\mathcal{I}'|$ is large. To enable a gradient-based attack, the server needs to make g be (twice-)differentiable w.r.t. $\mathcal{I}'$. Automatic differentiation (Autodiff) [9] can then be employed to calculate $\nabla_{\mathcal{I}'} \mathcal{L}_{atk}$. Users can still use the original g since

RAIFLE takes place on the server only. If g is a gradient-based learning algorithm, then our attack can also be considered "gradient matching" (eq. 1). Note that we do not require g to be differentiable w.r.t. θ for RAIFLE to work as long as g is differentiable w.r.t. $\mathcal{I}'$. Furthermore, under specific circumstances, RAIFLE is **convex** and can have a **unique global optimum** as we formally prove below:

Theorem 1 (Convexity of RAIFLE)**.** *Assume that g is twice-differentiable w.r.t. interactions $\mathcal{I}'$ and $\mathcal{L}_{atk}$ is the L_2 loss. If $\nabla^2_{\mathcal{I}'} g = \mathbf{0}$, then RAIFLE is convex w.r.t. $\mathcal{I}'$.*

Proof: Consider the gradient of $\mathcal{L}_{atk}$ w.r.t. any $\mathcal{I}'$:

$$\begin{aligned}\nabla_{\mathcal{I}'}\mathcal{L}_{atk}(\hat{\theta},\theta') &= \nabla_{\mathcal{I}'}||\hat{\theta} - g(\mathcal{X},\theta,\mathcal{I}')||_2^2 \\ &= -2\nabla_{\mathcal{I}'} g(\mathcal{X},\theta,\mathcal{I}') \cdot (\hat{\theta} - g(\mathcal{X},\theta,\mathcal{I}'))\end{aligned} \tag{6}$$

Thus, the Hessian of $\mathcal{L}_{atk}$ w.r.t. $\mathcal{I}'$ is:

$$\begin{aligned}&\nabla^2_{\mathcal{I}'}\mathcal{L}_{atk} \\ =& -2\nabla^2_{\mathcal{I}'} g(\mathcal{X},\theta,\mathcal{I}') \cdot \hat{\theta} + 2\nabla_{\mathcal{I}'} g(\mathcal{X},\theta,\mathcal{I}') \cdot \nabla^T_{\mathcal{I}'} g(\mathcal{X},\theta,\mathcal{I}') \\ =& 2\nabla_{\mathcal{I}'} g(\mathcal{X},\theta,\mathcal{I}') \cdot \nabla^T_{\mathcal{I}'} g(\mathcal{X},\theta,\mathcal{I}')\end{aligned} \tag{7}$$

Clearly, $\nabla^2_{\mathcal{I}'}\mathcal{L}_{atk} \succeq \mathbf{0}\ \forall \mathcal{I}' \in \mathbb{R}^m$ (i.e., positive semi-definite). We also know that $\mathcal{I}' \in \mathbb{R}^m$ is a convex set. Therefore, by the second-order convexity condition [15], RAIFLE is convex w.r.t. $\mathcal{I}'$. ■

From Eq. 7, we can see that (under the same conditions above) RAIFLE is strictly convex if and only if $\mathbf{rank}(\nabla_{\mathcal{I}'} g) = m$, and consequently, our attack can arrive at a unique solution that minimizes $\mathcal{L}_{atk}$. While making the learning algorithm g twice-differentiable w.r.t. $\mathcal{I}'$ is easy, requiring $\nabla^2_{\mathcal{I}'} g = \mathbf{0}$ is not always possible if the interactions "interact" with one another in g (see Section IV-A). To enable $\mathbf{rank}(\nabla_{\mathcal{I}'} g) = m$, the server can potentially manipulate the data (and the learning algorithm g) to keep the number of items m smaller than the number of training features d or model parameters p and remove any collinearity in the training features. It should be noted that even when the attack can provably converge to a unique optimal solution, there is no formal guarantee that the solution is the actual user interactions since we do not have any guarantee about the user's local updates $\hat{\theta}$.

C. Adversarial Data Manipulation

We further enhance the basic RAIFLE framework with a novel attack vector called Adversarial Data Manipulation (ADM), in which the FL server actively manipulates the item features presented to users to adversarially improve the reconstruction success rate. This technique is unique to IFL and does not apply to traditional FL since users in IFL interact with data prepared by the server. We present two specific ADM methods called *fingerprinting* and *noise injection*, along with a general approach for *indirect ADM* when direct control is not available.

1) Fingerprinting: In our first ADM method, the server modifies the item features to influence the training process such that the user's local update for any feature parameter can be deterministically zero or non-zero, thus acting as a form of signal or identifier for the server. Consider any single model parameter θ_j that is in direct contact with at least one feature of the training items. For example, if the model is linear or logistic regression, then any non-bias weight is one such parameter. Similarly, if the model is a neural network, then any non-bias weight in the first layer of the neural net will satisfy the criterion, while weights in other layers are only indirectly involved and thus do not fit the criterion. We refer to parameters that are in direct contact with the training features as *feature parameters* (see Figure 3).

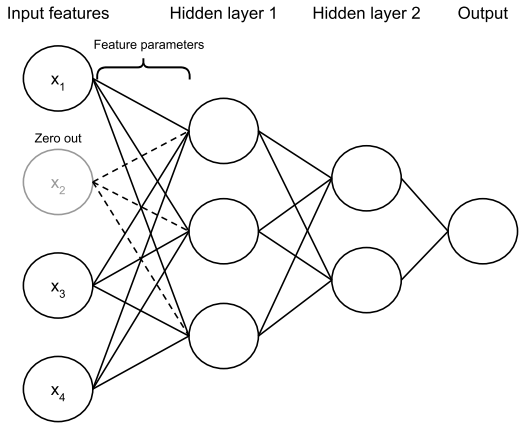

Fig. 3. Example of the fingerprinting method for a 2-layer neural net. The feature parameters consist of all connections between the inputs and the first hidden layer. Feature x_2 is zero-ed out, causing all feature weights corresponding to x_2 (dashed lines) to have 0 gradient during backpropagation.

Let F_{θ_j} denote the set of features that θ_j is in direct contact with, and let ∇_{θ_j} be the user's local update to θ_j via learning algorithm g. In order for ∇_{θ_j} to be non-trivial (i.e., non-zero), the values of the features in F_{θ_j} must also be non-trivial. If each feature in F_{θ_j} has the same values across all items, for example, then the model will not learn anything meaningful about the features since they do not contribute meaningfully to the model predictions. Thus, the server can deterministically control whether θ_j is updated or not by setting the values of the features in F_{θ_j} appropriately. More specifically, to make $\nabla_{\theta_j} \approx 0$, the server can simply set each feature in F_{θ_j} to exactly or near 0, while to make $\nabla_{\theta_j} \neq 0$, the server can simply keep the original feature values. By exclusively assigning certain features to specific items, the result of interacting with those items can be more easily determined from the model parameters updates. Hence, we call this technique *fingerprinting*.

2) Noise Injection: Although the fingerprinting idea with zeros described above is easy to demonstrate, it is not necessarily the best ADM method. In terms of execution, we would need to know how to assign features to each item and which features to set to zero such that the effect of interacting with one item can be isolated from all other items. In terms of detectability, the presence of zeros can be easily checked for, thus making the manipulation too obvious. In terms of performance, having too many zero-ed features can potentially reduce the magnitude of the gradient updates and consequently increase the difficulty of matching the gradients in the reconstruction. Through our experiments with gradient-based federated OLTR algorithms, we find that the reconstruction is

consistently successful when the feature values are replaced completely with random noise (e.g. Gaussian noise with a diagonal covariance matrix or uniform noise), especially in the case of neural net models. This phenomenon can be attributed to the following reasons: 1) each item's impact on the final gradients will be more "visible" due to the uniqueness of the item's features, thus facilitating the reconstruction; and 2) neural nets have a great capacity for learning from random noise and can even memorize them [6]. Using random noise also has two major advantages compared to the fingerprinting method: firstly, we do not need any complicated procedure to generate noise, and secondly, it is less detectable than having completely zero-ed out features. We call this method *noise injection*.

3) Indirect Manipulation: In certain scenarios, the FL server cannot arbitrarily alter the training feature values however it wants. While lacking direct control, the server can still indirectly influence the extracted features by modifying the representation of the items. We focus on the case where the features of the items are extracted locally by the users using a fixed procedure agreed upon by all participants. For example, in an image recommendation system, the server can only present images to users, and the image features can only be extracted using some pre-trained computer vision model for object classification. However, the server can still execute the noise injection method by adding small perturbations to the images such that the extracted features closely resemble the target noise [65]. The success of this method highly depends on how pliable the feature extractor is. Our experiments with image-based ranking show that existing computer vision models are highly susceptible to our adversarial manipulation, which leads to improved reconstruction performance (Section VI-C). We further enhance the reconstruction success by integrating the fingerprinting technique (Section IV-C).

To summarize, we present the high-level pseudocode for RAIFLE with ADM in Algorithm 1. Details about specific ADM implementations can be found in Section IV-C and V.

Algorithm 1 RAIFLE with ADM via noise injection

Input: Learning algorithm g, global FL model parameters θ, representations of items $\mathcal{X}$
Output: Reconstructed interactions $\mathcal{I}'$

```
// ADM stage
Determine the optimal ADM noise distribution via search
X' ← Modify X to induce the desired noise in training features
// FL Stage
Send θ and X' to a user for FL training
θ̂ ← Updated model parameters from user
// Reconstruction stage
I' ← Random initial interactions
while termination conditions are not met do
    θ' ← g(X', θ, I')
    ∇_{I'} ← Autodiff L_atk(θ̂, θ') w.r.t. I'
    Update I' using ∇_{I'}
end while
return I'
```

IV. Implementation Details

We describe important implementation details for RAIFLE, including how to enable automatic differentiation and choices of optimization algorithms and loss functions.

A. Enabling Differentiation w.r.t. Interactions

Typically, user interactions in RS/OLTR are treated as discrete values (e.g., clicked or not clicked). Furthermore, certain RS/OLTR algorithms might not be immediately differentiable w.r.t. the interactions $\mathcal{I}'$. To enable RAIFLE in these cases, the server needs to modify the learning algorithm g. First, the optimization algorithms employed by g must be differentiable. Furthermore, g needs to treat discrete interaction values as continuous "degrees" of interactions, with any non-differentiable usage of the interactions to be replaced with differentiable operations. We outline some general methods to modify representative RS/OLTR learning paradigms [45]:

Pointwise: If $\mathcal{L}_{FL}$ is not differentiable w.r.t. to $\mathcal{I}'$ (e.g. 0-1 classification loss), change it to a differentiable one such as L_2 loss. In fact, if $\mathcal{L}_{FL}$ is L_2 loss, then we can show that $\nabla^2_{\mathcal{I}'} g = \mathbf{0}$ (see Appendix A-A), which implies that the attack is convex (Theorem 1).

Pairwise: Instead of only including item pairs in which one item is "preferred" over the other (e.g. pairs of click and no click), $\mathcal{L}_{FL}$ can assign a weight to each pair based on how large the difference in preference is, then calculate the weighted sum of the losses of all pairs (see example in Section V-B3). While this can enable once-differentiation, the Hessian of g will likely not be equal to $\mathbf{0}$ since we now consider how user interactions "interact" with one another.

Listwise: There is a wide variety of listwise losses developed in the learning-to-rank literature [45], some of which are naturally differentiable w.r.t. user interactions, while for many others the relationship is not immediately clear without redefining the loss formulas. As in the case of pairwise losses, even though once-differentiation might be possible, $\nabla^2_{\mathcal{I}'} g$ might not be equal to $\mathbf{0}$. Due to the vast assortments of listwise methods and the lack of federated listwise algorithms, we leave the investigation of listwise attacks for future research.

B. Optimization Algorithms and Losses

There are various options for the loss function $\mathcal{L}$ and the optimization algorithm of RAIFLE. Previous research in gradient inversion attacks has often used the L_2 loss combined with the L-BFGS optimizer [86] or the cosine distance loss with the Adam optimizer [23]. We used the L_2 loss with the default L-BFGS and Adam optimizers from PyTorch [54] in our experiments, and found that while both optimizers can yield good results, PyTorch's L-BFGS is typically faster but needs to have its termination criteria appropriately set based on the magnitude of the loss. As such, we used L-BFGS for the non-image scenarios since it works without much tuning and Adam for the image scenarios since it yields better results than un-tuned L-BFGS. With regards to the loss, L_2 works well in all of our scenarios, but the loss values may need to be scaled up depending on the learning rate used in the local FL training for the L-BFGS optimizer to work properly. We simply divided the input gradients by the local FL learning rate in this case.

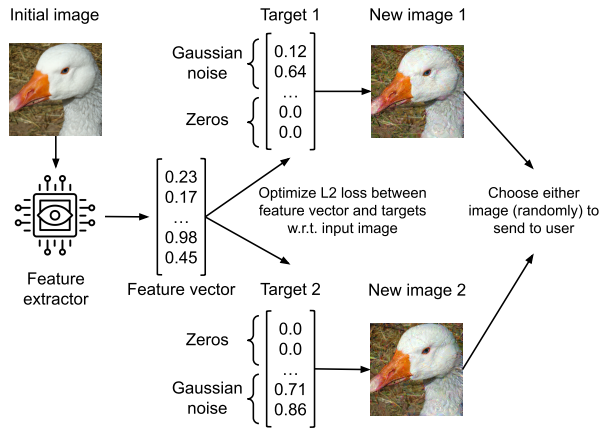

Fig. 4. Diagram of our partitioned noise injection ADM method for images. The FL server prepares two manipulated versions of an image by matching the image's extracted features to two different target noise vectors.

C. ADM with Noise Injection

The distributional characteristics of the noise used in the ADM noise injection method (Section III-C2) need to be set appropriately to maximize reconstruction success. The FL server can determine this empirically by simulating RAIFLE with its own data and fake interactions and performing a search of the noise's distributional parameters. In our experiments, we performed a simple grid search and found that Gaussian noise with mean 0 and standard deviation 0.1 and 4.0 consistently yields good results for the tabular and image LTR experiments, respectively. Other zeroth-order search algorithms such as evolution strategies [66] can also be employed to find the most suitable noise distribution.

In the case of image-based OLTR with a pre-trained feature extractor, we used the default Adam optimizer with a learning rate of 0.01 and 500 epochs to optimize the image inputs such that the L2 loss between the features extracted and a target Gaussian noise vector is minimized. As the last layer in the tested vision models is the ReLU activation which only outputs non-negative values, we clip the target noise to have a minimum of 0. Inspired by the fingerprinting technique, we also partitioned the target noise in half to create two different target vectors, with the first vector having the first half zero-ed out and the second vector having the second half zero-ed out (Figure 4). Using the two resulting manipulated images, the FL server can now randomly choose which version to send to each user. This *partitioned noise injection* method allows us to minimize the loss much better than with a single noise target, as higher numbers of extracted features generally make the optimization problem more difficult.

D. Software

We use PyTorch [54] to implement RAIFLE and the FL models for our experiments. We use IBM's Diffprivlib Python library [31] for the DP noise as it supports the Analytical Gaussian mechanism [8] for privacy budget $\varepsilon > 1$. For adversarial perturbation attacks, we use the Foolbox library [59], [60]. For differentiable optimizers, we use the Torchopt library [61].

Algorithm 2 RAIFLE for Federated RS (No ADM)

Input: Learning algorithm g, global model parameters θ, item embeddings $\mathcal{X}$, user's updated item embeddings $\hat{\mathcal{X}}$, user's updated model parameters $\hat{\theta}$
Output: Reconstructed interactions $\mathcal{I}'$, user embedding e'_u

$\mathcal{I}' \leftarrow$ Random initial interactions
$e'_u \leftarrow$ Random initial user embedding
while termination conditions are not met **do**
  $\mathcal{X}', \theta' \leftarrow g(\mathcal{X}, \theta, e'_u, \mathcal{I}')$
  $\nabla_{\mathcal{I}'}, \nabla_{e'_u} \leftarrow$ Autodiff $\mathcal{L}_{atk}(\hat{\theta}, \theta') + \mathcal{L}'_{atk}(\hat{\mathcal{X}}, \mathcal{X}')$ wrt $\mathcal{I}'$, e'_u
  Update $\mathcal{I}'$ and e'_u using $\nabla_{\mathcal{I}'}$ and $\nabla_{e'_u}$
end while
return $\mathcal{I}'$ and e'_u

V. Experiment Setup

Here, we describe our experiment setup for three evaluation scenarios: federated RS, federated OLTR with non-image data, and federated OLTR with image-based data.

A. Federated RS

1) Scenario: We focus on the collaborative filtering recommendation paradigm, specifically the FNCF algorithm [55]. As mentioned in Section II-C2, users in FNCF keep a private user embedding e_u that is not known to the FL server. This private embedding together with the public item embeddings $\mathcal{X}$ (known to the server) form the inputs to the global FL model to produce a personalized ranking score for each recommendation item. In addition to the learned model parameters $\hat{\theta}$, users also share the updated item embeddings $\hat{\mathcal{X}}$ with the server. The dimension of both the private and public embedding is 64. The FL model is a 3-layer neural net with 128, 64, and 32 hidden units and ReLU activation. For each user, we randomize the model parameters and user embedding, then train using the Adam optimizer for 20 epochs with a learning rate of 0.001. These choices are similar to our baseline (Section V-A4).

2) Data: We use MovieLens-100K [27], a widely used dataset in the RS literature, and the Steam-200K dataset [69] in the IMIA paper (Table II). To create the interaction labels for our experiment, we binarize the interactions between each user and each recommendation item (1 if interacted, 0 if not). For the local FL training of each user, we randomly sampled non-interacted items with a ratio of 4:1 to interacted ones.

TABLE II. Some statistics on federated RS datasets

Dataset	# of users	# of items	Avg. interactions per user
MovieLens-100K	943	1,682	106.0
Steam-200K	12,393	5,155	10.4

3) Attack: To account for the use of private user embedding and public item embeddings, we modify Algorithm 1 to perform joint optimization of user interactions and private user embedding (Algorithm 2), with the objective function being the sum of the model parameters loss and the item embedding loss (i.e., average L_2 distance between each pair of original and updated item embedding). Note that we do not apply ADM

in this scenario since the server receives an embedding for each item interacted with, which allows for (almost) perfect reconstruction of user interactions (Section VI).

4) Baseline: We use the IMIA method [82] as the comparison baseline for this scenario as it is the only work (to our knowledge) that attempts to reconstruct user interaction in the federated RS scenario. Note that IMIA is a stochastic heuristics search that does not make use of the gradient information and is limited to federated RS only. We only compare the FNCF algorithm as the federated graph recommendation approach tested in the IMIA paper does not fully specify how to protect the privacy of the graph. We also evaluate against IMIA's proposed regularization-based defense, which penalizes the distance between the updated item embeddings and the global embeddings using the L_1 loss.

B. Federated OLTR (non-image)

1) Scenario: We target the FPDGD algorithm [74] which is a pairwise method to OLTR. The FL server directly shares the training features with the users. Users perform multi-batch stochastic gradient descent (SGD) locally, where each SGD batch corresponds to one query and the associated items (no more than 10, sampled according to the scores of the ranking model similar to the FPDGD paper) and the local learning rate is 0.1. For the global FL model, we use a linear regression model and various neural nets with an increasing number of hidden units (followed by ReLU activation) to check the effect of having more parameters on reconstruction.

2) Data: We use the LETOR 4.0 dataset and the MSLR dataset used in the FPDGD paper, specifically the training set of the first fold of MQ2007 and MQ2008 in LETOR and of MSLR-WEB10K in MSLR [56] (Table III). We preprocessed the features in MSLR by normalizing them to have mean 0.0 and standard deviation 1.0. We simulated the user clicks using the "navigational" and "information" click chain model following prior work on federated OLTR [39], [74].

TABLE III. SOME STATISTICS ON OUR NON-IMAGE FOLTR DATASETS

Name	Features	Queries	Avg. # of items per query
MQ2008	46	471	20.4
MQ2007	46	1017	41.5
MSLR-WEB10K	136	6000	120.5

3) Attack: We employ RAIFLE with the noise injection ADM method. The FL server in this case will replace all of the training features with random noise drawn from the Gaussian distribution with mean 0.0 and standard deviation 0.1. To make FPDGD be differentiable w.r.t. the interactions, we utilize the technique described in Section IV-A for pairwise loss, specifically using $\mathcal{I}(1-\mathcal{I}^T)$ as the weight matrix.

4) Baseline: To our knowledge, no other work has attacked the federated OLTR scenario. As such, we compare our attack with the regular gradient inversion method (Section II-A), which can also be viewed as RAIFLE without ADM.

C. Federated OLTR (image-based)

1) Scenario: We simulate an image ranking scenario, where the FL server sends images to users who then "upvote" and "downvote" the images. The local FL training is done by first extracting features from the images via an object classification vision model and then performing 5 epochs of gradient descent with a learning rate of 0.01 on a linear or neural ranking model to minimize the pointwise ranking loss (Section IV-A). Compared to the previous scenarios, this setting is more difficult as the server no longer has direct control of the training features. Furthermore, any manipulation should be visually subtle to avoid being detected by human eyes. We experiment with several distinct computer vision models pre-trained on ImageNet to test the generalizability of our indirect noise injection technique across different architectures and feature dimensions (see Table IV). All of these models have a small number of parameters yet high accuracy on ImageNet, which makes them ideal for the FL setting where computation resources for deep learning can be constrained by the available hardware. We extract the representations from the last layer before the final classification layer as the ranking features.

TABLE IV. SOME STATISTICS ON TESTED COMPUTER VISION MODELS

Model	Number of parameters	Dimension of extracted features
ResNet18 [29]	11.7 mil	512
RegNet Y 800MF [57]	6.4 mil	784
DenseNet121 [32]	8.0 mil	1024
MNasNet 1.3 [71]	6.3 mil	1280

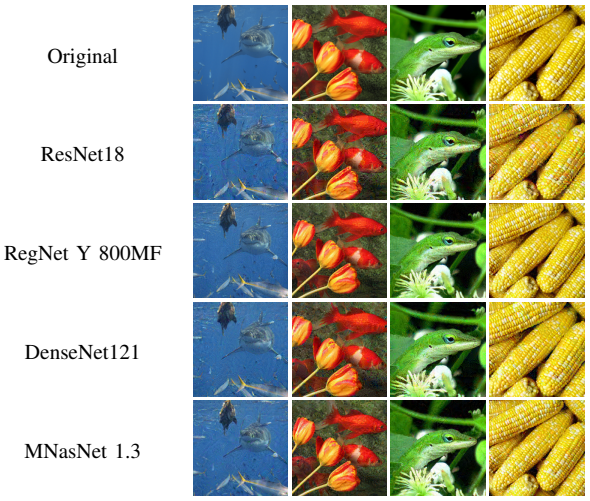

Fig. 5. Examples of original and manipulated images from ImageNet for different vision models. Some artifacts are visible but subtle.

2) Data: We use the ImageNet-1K (2012) dataset [64], particularly the validation split which consists of 50,000 images equally spread across 1,000 different classes. Each image is preprocessed using the default transformation in PyTorch's pre-trained vision models (i.e., resized and centrally cropped to size 224×224 then normalized via ImageNet standardization).

3) Attack: We use the partitioned noise injection method (Section IV-C) to manipulate each image for each target vision model. After performing ADM on an image, we undo the ImageNet normalization, convert the pixel values to 8-bit unsigned integers, and save the image in PNG format (about 147kB each). This results in a negligible difference of about 10^{-5} to 10^{-4} mean L_2 error in the preprocessed pixel values

before and after saving (Figure 5). To create user interactions, we randomly select each image as interacted with probability 0.5. To perform the reconstruction attack, we use the Adam optimizer with a learning rate of 0.1 running for 300 epochs.

4) Baseline: Similar to the non-image federated OLTR scenario, we use the vanilla gradient inversion without ADM as the baseline. Additionally, we test the Fast Gradient Sign Method (FGSM) [25] which perturbs the images to cause the models to misclassify by adding ϵ times the signs of the gradients of the classification loss w.r.t. the pixels ($\epsilon = 0.1$).

VI. Evaluation Results

In this section, we present our evaluation results for the scenarios described in Section V. We primarily report the *area under the curve* of the receiver operating characteristic (AUC) [16] since we do not want to rely on a fixed threshold for the classification and we equally care about positive and negative interactions. Note that randomly guessing will result in an AUC of ≈ 0.5. We summarize some highlights of our findings as follows:

- RAIFLE consistently outperforms baselines such as IMIA for federated RS and gradient inversion for both federated OLTR scenarios, achieving 0.8-1.0 AUC in most cases.
- RAIFLE achieves superior performance even when the server does not have direct control of the training features, particularly in the case of image-based federated OLTR.
- RAIFLE scales better than other techniques when the ratio of items to features increases.
- RAIFLE can better utilize an increasing number of model parameters than other techniques.

A. Federated RS

Table V presents RAIFLE's reconstruction performance on the FNCF algorithm for the Movie Lens-100K and Steam-200K datasets. We executed RAIFLE on each user exactly once. RAIFLE is able to achieve near-perfect reconstruction in this scenario even when ADM is not used. This can be explained by the fact that users share with the FL server an item embedding for each item, thus allowing the effect of each interaction to be easily determined. In fact, it is not necessary to use the model parameters for the reconstruction, as the item embeddings by themselves are sufficient.

Compared to the IMIA method, RAIFLE has significantly better performance, achieving > 0.9 F1 scores on both MovieLens-100K and Steam-200K (Table VI). With IMIA's proposed defense applied (L1 regularization term set to 1.0), RAIFLE still manages to achieve higher F1 scores than IMIA. Our results thus demonstrate the superiority of gradient-based optimization over a heuristic search like IMIA. Note that we can only compare the two methods using the F1 scores since IMIA only outputs a hard label.

TABLE V. RAIFLE's AUC on MovieLens-100K and Steam-200K

Dataset	Mean	Median	Standard deviation
MovieLens-100K	0.998	1.000	0.003
Steam-200K	0.960	1.000	0.190

TABLE VI. Avg. F1 scores (threshold 0.5) for RAIFLE and IMIA (from [82]). RAIFLE outperforms IMIA in both datasets by > 0.1 with IMIA defense and > 0.3 without.

Method	IMIA Defense	MovieLens-100K	Steam-200K
IMIA	No	0.593	0.671
RAIFLE	No	**0.983**	**0.923**
IMIA	Yes	0.215	0.206
RAIFLE	Yes	**0.382**	**0.316**

B. Federated OLTR (non-image)

TABLE VII. Mean reconstruction AUC for FPDGD on MQ2007 and MSLR-WEB10K. RAIFLE outperforms vanilla gradient inversion in all scenarios by a significant margin.

MQ2007

Model	ADM	Informational			Navigational		
		4	8	16	4	8	16
Linear	None	0.87	0.76	0.66	0.95	0.84	0.71
	RAIFLE	**1.00**	**0.98**	**0.82**	**1.00**	**0.98**	**0.88**
Neural (4 hid. units)	None	0.78	0.66	0.57	0.86	0.70	0.60
	RAIFLE	**1.00**	**0.99**	**0.95**	**1.00**	**1.00**	**0.98**
Neural (8 hid. units)	None	0.77	0.64	0.56	0.86	0.68	0.59
	RAIFLE	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**
Neural (16 hid. units)	None	0.75	0.60	0.53	0.82	0.61	0.55
	RAIFLE	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

MSLR-WEB10K

Model	ADM	Informational			Navigational		
		12	24	48	12	24	48
Linear	None	0.54	0.51	0.50	0.55	0.51	0.51
	RAIFLE	**1.00**	**0.98**	**0.80**	**1.00**	**0.97**	**0.86**
Neural (4 hid. units)	None	0.52	0.50	0.50	0.52	0.51	0.50
	RAIFLE	**0.98**	**0.95**	**0.87**	**0.99**	**0.97**	**0.93**
Neural (8 hid. units)	None	0.51	0.50	0.50	0.51	0.50	0.50
	RAIFLE	**0.99**	**0.96**	**0.91**	**1.00**	**0.99**	**0.96**
Neural (16 hid. units)	None	0.51	0.50	0.50	0.51	0.50	0.51
	RAIFLE	**0.97**	**0.94**	**0.91**	**0.99**	**0.98**	**0.96**

Table VII showcases RAIFLE's performance for the FPDGD algorithm on the MQ2007 and MSLR-WEB10K dataset (we omit MQ2008 since its result is very similar to MQ2007). We tested each combination of ranking model and click model with 4, 8, and 16 total queries for the multibatch local training on MQ2007, and 12, 24, and 48 queries on MSLR-WEB10K. These numbers of queries are intentionally chosen to scale with the number of available features in each dataset (e.g. 4 queries and 12 queries result in nearly the same number of items as the number of features in MQ2007 and MSLR-WEB10K, respectively, while 8, 16 and 24, 48 are double and quadruple). We can see that RAIFLE with our noise injection ADM method has much better AUC than RAIFLE without ADM (i.e., vanilla gradient inversion), achieving > 0.9 AUC in all scenarios except for the linear ranker with 16 queries setting, which is also the most difficult since it has the least number of model parameters and the most number of items. Particularly, on the MSLR-WEB10K dataset, the reconstruction without ADM is barely better than random guessing. Furthermore, RAIFLE with ADM is more capable of utilizing the increasing number of model parameters to attain better reconstruction results than without ADM.

TABLE VIII. MEAN RECONSTRUCTION AUC (ROUNDED TO 3RD DECIMAL PLACE) FOR FEDERATED OLTR ON IMAGE-BASED DATA. THE 1X, 2X, AND 4X UNDERNEATH THE RANKING MODEL TYPE REFER TO THE RATIO OF IMAGES TO FEATURE DIMENSIONS. EACH CONFIGURATION WAS RUN 200 TIMES WITH RANDOMLY SAMPLED MODEL PARAMETERS AND INTERACTIONS. RAIFLE OUTPERFORMS BASELINES IN MOST SCENARIOS.

Vision Model	ADM	Linear			Neural (2 hidden units)			Neural (4 hidden units)			Neural (8 hidden units)		
		1x	2x	4x	1x	2x	4x	1x	2x	4x	1x	2x	4x
ResNet18 (512 features)	None	1.000	0.916	0.767	0.921	0.868	0.771	0.985	0.945	0.857	0.999	0.985	0.924
	FGSM	1.000	0.915	0.766	0.914	0.857	0.759	0.981	0.934	0.841	0.998	0.977	0.906
	RAIFLE	**1.000**	**0.943**	**0.772**	**0.946**	**0.920**	**0.823**	**0.993**	**0.981**	**0.922**	**1.000**	**0.998**	**0.978**
RegNet Y 800MF (784 features)	None	0.999	0.918	**0.772**	0.948	0.913	0.801	0.991	0.977	0.891	1.000	0.994	0.925
	FGSM	0.999	0.913	0.771	0.948	0.908	0.796	0.990	0.974	0.884	1.000	0.993	0.922
	RAIFLE	**1.000**	**0.952**	0.767	**0.956**	**0.932**	**0.833**	**0.993**	**0.989**	**0.938**	**1.000**	**0.999**	**0.984**
DenseNet121 (1024 features)	None	**0.933**	**0.772**	**0.667**	0.902	0.805	0.700	0.970	0.896	0.770	0.995	0.946	0.827
	FGSM	0.933	0.771	0.666	0.891	0.794	0.691	0.961	0.884	0.759	0.992	0.932	0.810
	RAIFLE	0.919	0.765	0.664	**0.923**	**0.825**	**0.717**	**0.983**	**0.935**	**0.815**	**0.999**	**0.983**	**0.904**
MNasNet 1.3 (1280 features)	None	0.994	0.895	0.764	0.936	0.858	0.754	0.985	0.926	0.787	0.996	0.942	0.775
	FGSM	0.989	0.885	0.758	0.930	0.845	0.742	0.982	0.913	0.771	0.994	0.928	0.755
	RAIFLE	**1.000**	**0.939**	**0.775**	**0.940**	**0.882**	**0.791**	**0.991**	**0.965**	**0.875**	**0.999**	**0.992**	**0.924**

C. Federated OLTR (image-based)

Table VIII presents the mean AUC and standard deviation of RAIFLE and baselines for each vision model, ranker model (linear and neural net with 2, 4, and 8 hidden units, ReLU activation), and ratio of images to feature dimensions (1x, 2x, and 4x). Overall, we can observe that RAIFLE consistently outperforms vanilla gradient inversion and FGSM across most settings. The only case where RAIFLE is not consistently better is the linear ranker scenario, although RAIFLE is still within 0.02 AUC from the best method. As the number of images increases, the reconstruction becomes harder overall, but RAIFLE is least affected. The FGSM method does not improve upon vanilla gradient inversion, which indicates that perturbing the images to cause misclassification does not automatically extend to our problem of reconstructing interactions.

VII. COUNTERMEASURES

In this section, we look at several possible countermeasures for RAIFLE, including local differential privacy (LDP) and secure aggregation (SA). See Appendix A-C for more discussion.

A. Local Differential Privacy

To test the effect of LDP on RAIFLE, we experiment with the Gaussian mechanism by applying Gaussian noise to each user's local update using privacy budget $\varepsilon \in \{1, 20, 100, 500\}$ ($\delta = 10^{-8}$). Table IX shows the results of applying LDP on two example configurations from each evaluation scenario in Section V (FNCF with ML-100K and Steam-200K for federated RS; FPDGD with linear ranker, informational click model, and 16 queries on MQ 2007 and MSLR-WEB10K; neural ranker with 8 hidden units and number of items equal to number of features using ResNet18 and DenseNet121). ε is chosen to represent a wide range of privacy protections, with $\varepsilon = 1$ providing particularly strong guarantees under LDP (at the expense of utility) and $\varepsilon = 500$ representing a typical LDP budget adopted in LDP research and applications [11], [28]. We set the sensitivity $\Delta_2 = 0.1, 0.5, 0.05$ for federated RS, FPDGD, and FOLTR with images, respectively. We use the Analytical Gaussian formula [8] for $\varepsilon > 1$ and the standard one for $\varepsilon = 1$ [21].

TABLE IX. MEAN RECONSTRUCTION AUC OF SOME REPRESENTATIVE EVALUATION SCENARIOS WITH LDP APPLIED.

Scenario	ADM	$\varepsilon = 1$	$\varepsilon = 20$	$\varepsilon = 100$	$\varepsilon = 500$	No DP
FNCF w/ ML-100K	N/A	0.50	0.52	0.56	0.74	1.00
FNCF w/ Steam-200K	N/A	0.50	0.56	0.72	0.90	0.96
FPDGD w/ MQ 2007	None	0.50	0.54	0.57	0.58	0.66
	RAIFLE	0.50	0.56	0.66	0.75	0.82
FPDGD w/ MSLR10K	None	0.50	0.50	0.50	0.50	0.50
	RAIFLE	0.50	0.52	0.58	0.62	0.80
FOLTR w/ ResNet18	None	0.50	0.52	0.55	0.62	1.00
	RAIFLE	0.50	0.52	0.55	0.63	1.00
FOLTR w/ DenseNet121	None	0.50	0.51	0.54	0.59	1.00
	RAIFLE	0.50	0.51	0.53	0.59	1.00

TABLE X. FPDGD'S AVG. TEST NDCG@10 VS LDP ε (MSLR-10K).

Model	Click model	$\varepsilon = 1$	$\varepsilon = 20$	$\varepsilon = 100$	$\varepsilon = 500$	No DP
Linear	Informational	0.228	0.280	0.298	0.301	0.315
Linear	Navigational	0.225	0.294	0.310	0.309	0.328
Neural	Informational	0.223	0.226	0.218	0.228	0.282
Neural	Navigational	0.227	0.243	0.256	0.263	0.291

Overall, we see that LDP can reduce RAIFLE's effectiveness close to random guessing, but only with sufficiently small ε. At $\varepsilon = 500$, while the reconstruction performance is decreased across all scenarios and methods, for FNCF, FPDGD on MQ2007, and FOLTR with images, the performance remains relatively competitive. Non-ADM and ADM's performance in the two image-based configurations is essentially the same, but in the FPDGD case, ADM is slightly better. This higher performance from ADM is not because ADM can overcome LDP, but rather because without ADM in those cases, the reconstruction would not even work.

Despite the capability of LDP, it can significantly reduce the utility of the learned FL model [78]. To illustrate the privacy-utility tradeoff, we simulate the FL training for the FPDGD scenario on the MSLR-WEB10K dataset with a linear model and a neural net with 16 hidden units. The FL model is trained on a single pass of the train portion in Fold 1 of the dataset, where each user is assigned one unique query.

FL aggregation occurs for every 100 users. We measure the normalized discounted cumulative gain for the top 10 items (NDCG@10) on the corresponding test portion. The simulation is repeated 10 times. Our results show that even for $\varepsilon = 100$, the FL model's performance suffers a noticeable decrease (Table X). Thus, FL practitioners often need to use large LDP $\varepsilon > 10$ to achieve practical model performance [11], [28]. Determining the right balance between privacy protection and application utility is essential to adequately defend against attacks like RAIFLE. For more results on the privacy-utility tradeoff of FNCF and FPDGD, we refer readers to Table 4 in the IMIA paper [82] and Table 2 in the FPDGD paper [74].

B. Secure Aggregation with DP

Due to the high utility impact of LDP, a more effective defense is to combine SA and DP to reduce the amount of noise needed while achieving a similar level of privacy [34]. Although this approach can hide the link between user updates and their identities, it is not invulnerable to our ADM technique. The FL server can single out the local update for a target user u_k from the securely aggregated result by executing our fingerprinting technique, optionally in combination with noise injection (Figure 6):

1) Choose a subset D of the training features.
2) For target user u_k: set all features $\notin D$ to 0. (Optional: Inject noise into all features $\in D$).
3) For all other users: set all features $\in D$ to 0.
4) Send the modified features to users for the FL training.

User	d_1	d_2	d_3	d_4	d_5	d_6
⋮	⋮	⋮	⋮	⋮	⋮	⋮
u_{k-1}	0.1	0.2	0.3	**0.0**	**0.0**	**0.0**
$\boldsymbol{u_k}$	**0.0**	**0.0**	**0.0**	0.4	0.5	0.6
u_{k+1}	0.1	0.2	0.3	**0.0**	**0.0**	**0.0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig. 6. Example of bypassing vanilla SA with 6 training features via our fingerprinting ADM technique. Target user u_k has features d_1, d_2, d_3 set to 0 while features d_4, d_5, d_6 are non-zero.

As described in Section III-C, the local update from user u_k will be non-zero for the feature parameters in direct contact with D and will be zero for all other feature parameters, while for all other users, the reverse applies. Thus, when the local updates from all users are securely aggregated (e.g. via summation), the final result will leak u_k's local update since we are only adding 0's from other users to u_k's update. Consequently, the server can now execute RAIFLE on user u_k using the singled-out updates.

To illustrate the performance of RAIFLE against SA, we simulate SA with Gaussian LDP noise and a varying number of FL participants n in the FPDGD scenario on the MQ2007 dataset. Using the above technique with noise injection, we randomly choose one target user u_k and let D be the entire feature space, thus setting the gradients of all other users to $\mathbf{0}$. Effectively, the final aggregated result is equal to:

$$\nabla_{u_k} + N(0, n\sigma_\varepsilon^2)$$

where σ_ε is the noise scale corresponding to LDP ε. From Table XI, we see that compared to using LDP alone, SA with LDP can better protect user privacy against RAIFLE with a sufficiently large number of participants, but it does not completely prevent leakage.

TABLE XI. RAIFLE'S AVERAGE AUC ON FPDGD FOR MQ2007 WITH SA AND LDP (4 QUERIES, INFORMATIONAL CLICK MODEL)

Model	ε	Number of participants			
		10	100	500	1000
Linear	∞	1.00	1.00	1.00	1.00
	700	0.79	0.64	0.56	0.54
	500	0.75	0.60	0.55	0.54
	300	0.71	0.57	0.54	0.53
	100	0.61	0.54	0.52	0.51
Neural (16 hidden units)	∞	1.00	1.00	1.00	1.00
	700	0.81	0.63	0.56	0.54
	500	0.77	0.60	0.54	0.53
	300	0.73	0.58	0.52	0.51
	100	0.61	0.54	0.51	0.51

Note that this attack is not restricted to just a single target user. By allocating a distinct portion of the feature space to each target user and making sure that only each such user can have non-zero values for their allocated features, the server can learn their individual update from the final aggregated result by looking at the corresponding feature parameters. However, this also reduces the number of available features for reconstruction and will likely affect the attack performance. If the server can choose the participants, then this attack can also be combined with Sybil-based attacks [12] to remove all DP noise from other users, effectively getting rid of SA.

C. Detecting Data Manipulation

1) Cryptography: To prevent the server from executing ADM, users can validate the integrity and authenticity of the items sent by the server using cryptography techniques. For example, users can compare cryptographic hashes (i.e., checksum) of the items to ensure that their training features are the same as all of their peers (similar to the parameters validation defense for model manipulation in [53]). However, different users can have different items, which renders the process of cross-checking items more difficult, not to mention the need to do so in a privacy-preserving manner to avoid leaking information to other users. Furthermore, detecting inconsistency among users would fail if the server manipulates the data in the same manner for all users.

2) Heuristic Checks: One possible ADM detection method is to analyze how it affects the resulting FL gradients. To visualize the effects, we use the t-SNE method [73] with two components and varying perplexity on the non-image and image-based FOLTR scenarios. We see that ADM-impacted gradients exhibit some noticeable differences in the case of FPDGD on MSLR-WEB10K with a neural ranker (Figure 8). However, in the case of image-based FOLTR with ResNet18 using a neural ranker, we do not observe any obvious difference (Figure 9). This indicates that looking at the gradients alone does not necessarily tell us if the training data has been tampered with or not.

Another method is to directly check the training data for any sign of ADM. Detecting our fingerprinting method is

straightforward: simply checking for the presence of zeroed-out (or constant) features would suffice. Detecting noise injection, however, is more difficult: while we can try to quantify the degree of randomness in the data (e.g., via Shannon entropy), a malicious server can mask the noise, such as by mixing noise and real data via a convex combination. As another example, our image-based attack (Section IV-C) can include a regularizer to further reduce the visual artifacts (Appendix A-B). Designing a comprehensive set of data quality checks is likely not practically feasible due to the vast possibilities of manipulation techniques and data modalities. The difficulty of detecting active manipulation is also reflected in model manipulation attacks [13].

D. Minimizing Shared Information

While current FL schemes often rely on the sharing of gradients or parameters, such high-dimensional information can be easily exploited to breach user privacy as we have empirically shown in our paper. From our analysis of the implications of Theorem 1, the number of shared parameters must be greater than or equal to the number of items for the reconstruction to be able to find a unique solution. Thus, to make the reconstruction attack more difficult, one approach is to reduce the number of shared parameters or to share something else completely different. The FOLtR-ES algorithm [39] is one example federated OLTR algorithm that allows users to share with the FL server a single number for the local ranker's utility (using some locally sampled model parameters). In OLTR, such utility metric (e.g., discounted cumulative gain) is often not differentiable w.r.t. the interactions, thus preventing RAIFLE from working properly. While such gradient-free methods can potentially deter reconstruction attacks, they often have lower utility than gradient-based methods. Furthermore, DP noise is still necessary to guarantee theoretical privacy protection, although the amount needed might be lower than when the full model parameters are shared.

E. Personalization

Personalized FL (PFL) is an emerging FL approach that aims to learn a customized model for each user to address the issue of heterogeneous data as well as provide user-specific personalization [70]. In the context of federated RS, collaborative filtering-based methods such as FNCF can be considered PFL since they learn a private user embedding for each user. From a privacy point of view, PFL offers both opportunities and challenges: introducing a privately-learned component can potentially help users rely less on the global model, thus reducing the amount of information needed to share with the server, but at the same time, the private component does not guarantee complete privacy protection. Our experiment with federated RS demonstrates that the FL server can still achieve excellent reconstruction performance despite the hidden FNCF user embedding. Recently, there has been some attempt at personalizing the user's view of the data through private *item* embeddings [44] for collaborative filtering. This strategy could potentially prevent RAIFLE since the number of unknown parameters to estimate would be higher than the number of items. Overall, we believe PFL can be a promising approach towards privacy, although careful design is still recommended.

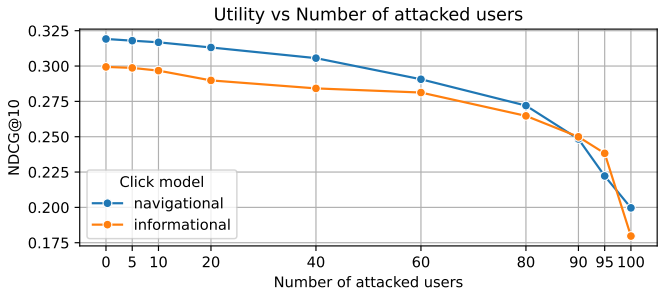

Fig. 7. Utility (test NDCG@10) vs number of attacked users (out of 100) for a linear FPDGD ranker on the MSLR-WEB10K dataset.

TABLE XII. PERCENTAGE OF MANIPULATED FEATURES VS MEAN RECONSTRUCTION AUC FOR FPDGD ON MQ2007 AND MSLR-WEB10K

Dataset	Model	Informational				Navigational			
		0%	50%	75%	100%	0%	50%	75%	100%
MQ2007 (16 queries)	Linear	0.65	0.77	0.79	0.82	0.71	0.86	0.87	0.88
	Neural 4	0.57	0.72	0.88	0.95	0.60	0.79	0.95	0.98
	Neural 8	0.56	0.71	0.91	0.99	0.58	0.78	0.98	1.00
	Neural 16	0.53	0.67	0.90	1.00	0.55	0.74	0.98	1.00
MSLR-10K (48 queries)	Linear	0.50	0.51	0.65	0.80	0.50	0.52	0.70	0.86
	Neural 4	0.50	0.50	0.54	0.87	0.50	0.50	0.55	0.93
	Neural 8	0.50	0.50	0.52	0.91	0.50	0.50	0.53	0.96
	Neural 16	0.50	0.50	0.51	0.91	0.51	0.50	0.51	0.96

VIII. DISCUSSION

We discuss RAIFLE's characteristics, current limitations, and possible future extensions.

A. Impact on FL Utility

While RAIFLE requires some modifications to the local learning algorithm (Section III-B), this modified version is only used by the server for performing the gradient matching optimization and is never used by real users, thus never affecting the FL model. The use of manipulated data for ADM, however, will certainly reduce the performance of the global FL model if not handled properly. Depending on the FL setup and the amount of control available, the server can take specific measures to limit the negative impacts of ADM. If SA is not applied, the server can simply disregard all FL updates from targeted users since their identities are known. Should discarding bad FL updates be infeasible, the server can choose a small number of users to apply ADM in one or a few FL rounds only instead of attacking every user in every round, thereby minimizing the low-quality contributions from targeted users. As a demonstration, from Figure 7, the test NDCG@10 for a linear FPDGD ranker trained on the MSLR-WEB10K dataset with 100 users is only reduced by $\approx$ 5-10% even when 20-40% of users have random item features. If SA is applied, ADM will have to be performed for every user (Section VII-B), thus requiring the server to discard or significantly reduce the weight of the affected FL rounds' results, which is achievable if the server can control each FL round and the participants.

B. Constrained Server Capabilities

In certain settings, the central server might be limited in its capability to manipulate the training features or even be

non-existent (as in the case of decentralized federated recommendation systems; see Appendix A-C3). Here, we consider a constrained scenario where the server can only control a subset of the features. We repeat the FPDGD experiment (Section V-B) but restrict to manipulating only 50% and 75% of all features. Our results from the hardest settings tested for both MQ2007 and MSLR-WEB10K (without DP) indicate a positive correlation between RAIFLE's attack performance and the percentage of manipulated features (Table XII). However, we can observe that the impact varies depending on the dataset and the number of items, with MSLR-WEB10K incurring a much steeper reduction in mean attack AUC compared to MQ2007, essentially no better than random guessing when manipulating only 50% of features. We can also see that the effect of increasing the number of model parameters on attack AUC is mostly reversed when manipulating $< 100\%$ of features, with more features leading to weaker reconstruction performance (except for MQ2007 at 75% manipulation). In the case where no features can be manipulated but the presentation of the items can be modified (e.g., ordering), it might be possible for the server to exploit users' click bias w.r.t. the item positions or popularity [82]. However, it is unclear how to turn this angle into an effective or efficient attack since we cannot readily leverage such discrete manipulation for optimization.

C. Computational and Storage Overhead

In our experiments, image-based manipulation is the only scenario where our attack needs significantly more computational resources. With a single Nvidia Tesla V100 16GB GPU, it takes ≈ 5 minutes to manipulate each batch of 256 images for DenseNet121, the largest model tested. Note that this step only needs to be done once since the manipulated images can be reused for different users. Performing the gradient inversion in this scenario is much faster: only ≈ 20 seconds for each user with 4096 images on DenseNet121. Considering the massive computing capabilities of adversaries like Google or Facebook, our attack is very cheap to perform, not to mention it can be executed separately from the FL protocol, thus avoiding any impact on the FL protocol's speed. With regards to storage space, in the case of images, we would need to additionally store at least one extra manipulated image for every original image. However, in the case of direct non-image manipulation, we can simply store a single random seed for each user so that we can recreate the attack noise vector anytime.

D. Relationship to Model Manipulation and Other Attacks

Although model manipulation [22], [53], [13] and RAIFLE's ADM method are both active attacks that lead to certain adversarial behaviors in users' local FL updates, they have several major differences in terms of techniques and applicability. First, model manipulation depends on the presence of some specific model architectural components such as ReLU activation or CNN layers, whereas ADM does not have any explicit requirements for the FL models (other than differentiability w.r.t. interactions). Second, achieving specific behaviors for the gradients is an explicit goal in model manipulation, but for data manipulation, the exact influence on the gradients is not always considered, as in the case of our noise injection method. Third, model manipulation may not be possible in certain scenarios, such as when users cryptographically collaborate to choose the initial global model and cross-check the aggregated result at each FL round (unless the server inserts Sybil users [12]). Achieving such consensus can be difficult with server-prepared data as users may have different views of the same items.

Aside from model manipulation, RAIFLE also shares some minor similarities with other ML attacks while still exhibiting noteworthy differences. RAIFLE has a similar goal to that of gradient inversion [84], [52] but focuses more on inverting the "labels" produced by the users. It involves modifying input data like the adversarial perturbation or backdoor attacks [3], [7], [41], although its aim is not to mislead models but to extract more information from users. RAIFLE thus can be considered a distinct blend of existing classes of ML attacks designed to attack IFL or similar scenarios. We believe our attack set a novel research direction in ML system vulnerabilities and exploits, particularly interactive ML systems where users do not have full control over their experiences.

E. Limitations and Future Work

RAIFLE is currently restricted to FL settings wherein a central server controls the features of user interaction items, with federated recommendation and learning-to-rank as the two most prominent examples. In this sense, RAIFLE is not as flexible as model manipulation attacks [22], [53], [13] which can be applied in more general (but still malicious) FL scenarios. However, we argue that federated RS and OLTR are among the most important applications of FL to study given how widely-used services like search engines, e-commerce product suggestions, and entertainment recommendations all collect a vast amount of sensitive user data. Furthermore, there have been several prominent real-world cases of large-scale recommendation data manipulation, with companies like Facebook, Amazon, and Google manipulating the new feeds or search recommendations for *hundreds of thousands* of users [50], [62], [5]. These examples highlight the relevance of our strong threat model and the genuine risk of centralized entities exploiting their positions for their own benefit.

With regards to RAIFLE's design, this work mainly explores numeric feature representations only and is not yet extensible to non-differentiable item representations such as natural text. User interactions are also currently restricted to values to which the reconstruction can be made differentiable. As such, we intend to develop ADM techniques for more types of item modalities, including multimodal representations (e.g., image and text), and to investigate more IFL scenarios in which the interactions cannot be easily "smoothened". Our proposed ADM techniques involve manipulating data only and were developed with knowledge of the underlying FL algorithm. It is potentially possible to combine data manipulation with model manipulation [22], [53], [13] to achieve a better attack. Automatic discovery of ADM, particularly for determining the best noise injection method, is also of special interest. Lastly, we intend to look into more covert ADM methods by introducing a manipulation budget while retaining strong reconstruction performance. Our preliminary results in this direction via restricting the number of manipulated features show a rather noticeable reduction in the attack's capability, which we hope to remediate. We leave these for future research.

IX. Conclusion

This work identifies and explores the threat of reconstruction attacks in interaction-based federated learning (IFL), particularly in the context of federated recommender systems (RS) and online learning to rank (OLTR). Our novel optimization-based RAIFLE method exploits the IFL server's knowledge and control over the interaction items to execute Adversarial Data Manipulation (ADM), a unique attack vector where the server actively manipulates the items. We demonstrate that RAIFLE can achieve superior reconstruction performances compared to existing attacks in a variety of settings. Our results shed light on the dire risk of inferring user interactions in IFL by malicious servers. We discuss a variety of countermeasures against RAIFLE and ADM, including differential privacy, secure aggregation, and manipulation detection.

Acknowledgment

We would like to thank Professor Negin Rahimi for her helpful comments on learning-to-rank and information retrieval. The project was supported in part by NSF grant 2131910.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[2] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2807385

[3] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in adversarial attacks and defenses in computer vision: A survey," *IEEE Access*, vol. 9, pp. 155 161–155 196, 2021.

[4] M. Ammad-ud-din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *CoRR*, vol. abs/1901.09888, 2019. [Online]. Available: http://arxiv.org/abs/1901.09888

[5] AP News, "Google loses final eu court appeal against 2.4 billion euro fine in antitrust shopping case," 2024. [Online]. Available: https://apnews.com/article/google-european-union-antitrust-shopping-court-a281e4e4722efa816e929a52a9939d86

[6] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 233–242. [Online]. Available: https://proceedings.mlr.press/v70/arpit17a.html

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[8] B. Balle and Y.-X. Wang, "Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm, Sweden: PMLR, 10–15 Jul 2018, pp. 394–403. [Online]. Available: https://proceedings.mlr.press/v80/balle18a.html

[9] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 5595–5637, jan 2017. [Online]. Available: https://www.jmlr.org/papers/volume18/17-468/17-468.pdf

[10] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1253–1269. [Online]. Available: https://doi.org/10.1145/3372297.3417885

[11] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2019. [Online]. Available: https://arxiv.org/pdf/1812.00984

[12] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "Reconstructing individual data points in federated learning hardened with differential privacy and secure aggregation," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2023, pp. 241–257. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/EuroSP57164.2023.00023

[13] ——, "When the curious abandon honesty: Federated learning is not private," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, 2023, pp. 175–199. [Online]. Available: https://doi.org/10.1109/EuroSP57164.2023.00020

[14] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1175–1191. [Online]. Available: https://doi.org/10.1145/3133956.3133982

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: https://doi.org/10.1017/CBO9780511804441

[16] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320396001422

[17] J. Canny, "Collaborative filtering with privacy," in *Proceedings 2002 IEEE Symposium on Security and Privacy*, 2002, pp. 45–57. [Online]. Available: https://doi.org/10.1109/SECPRI.2002.1004361

[18] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, vol. 36, no. 05, pp. 11–20, sep 2021. [Online]. Available: https://doi.org/10.1109/MIS.2020.3014880

[19] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, p. 965–981, nov 1998. [Online]. Available: https://doi.org/10.1145/293347.293350

[20] R. Cummings, D. Desfontaines, D. Evans, R. Geambasu, Y. Huang, M. Jagielski, P. Kairouz, G. Kamath, S. Oh, O. Ohrimenko, N. Papernot, R. Rogers, M. Shen, S. Song, W. Su, A. Terzis, A. Thakurta, S. Vassilvitskii, Y.-X. Wang, L. Xiong, S. Yekhanin, D. Yu, H. Zhang, and W. Zhang, "Advancing Differential Privacy: Where We Are Now and Future Directions for Real-World Deployment," *Harvard Data Science Review*, vol. 6, no. 1, jan 16 2024. [Online]. Available: https://hdsr.mitpress.mit.edu/pub/sl9we8gh

[21] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, p. 211–407, aug 2014. [Online]. Available: https://doi.org/10.1561/0400000042

[22] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," in *International Conference on Learning Representations*, September 2021. [Online]. Available: https://openreview.net/forum?id=fwzUgo0FM9v

[23] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3495724.3497145

[24] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using

collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, p. 61–70, dec 1992. [Online]. Available: https://doi.org/10.1145/138859.138867

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[26] A. Grotov and M. de Rijke, "Online learning to rank for information retrieval: Sigir 2016 tutorial," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1215–1218. [Online]. Available: https://doi.org/10.1145/2911451.2914798

[27] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, dec 2015. [Online]. Available: https://doi.org/10.1145/2827872

[28] F. Hartman and P. Kairouz, "Distributed differential privacy for federated learning," Mar 2023. [Online]. Available: https://ai.googleblog.com/2023/03/distributed-differential-privacy-for.html

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[30] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized recommendation based on matrix factorization: A comparison of gossip and federated learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 317–332. [Online]. Available: https://doi.org/10.1007/978-3-030-43823-4_27

[31] N. Holohan, S. Braghin, P. M. Aonghusa, and K. Levacher, "Diffprivlib: The IBM differential privacy library," *CoRR*, vol. abs/1907.02444, 2019. [Online]. Available: http://arxiv.org/abs/1907.02444

[32] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269. [Online]. Available: https://doi.org/10.1109/CVPR.2017.243

[33] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 7232–7241. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/3b3fff6463464959dcd1b68d0320f781-Paper.pdf

[34] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 5201–5212. [Online]. Available: http://proceedings.mlr.press/v139/kairouz21a.html

[35] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. [Online]. Available: http://dx.doi.org/10.1561/2200000083

[36] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam, and S. K. Jha, "De-anonymisation attacks on tor: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2324–2350, 2021. [Online]. Available: https://doi.org/10.1109/COMST.2021.3093615

[37] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008, pp. 531–540. [Online]. Available: https://doi.org/10.1109/FOCS.2008.27

[38] S. Y. Khamaiseh, D. Bagagem, A. Al-Alaj, M. Mancino, and H. W. Alomari, "Adversarial deep learning: A survey on adversarial attacks and defense mechanisms on image classification," *IEEE Access*, vol. 10, pp. 102 266–102 291, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3208131

[39] E. Kharitonov, "Federated online learning to rank with evolution strategies," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 249–257. [Online]. Available: https://doi.org/10.1145/3289600.3290968

[40] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: single database, computationally-private information retrieval," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*, 1997, pp. 364–373. [Online]. Available: https://doi.org/10.1109/SFCS.1997.646125

[41] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 5–22, 2024. [Online]. Available: https://doi.org/10.1109/TNNLS.2022.3182979

[42] Y. Li, H. Yu, Y. Zeng, and Q. Pan, "Hfsa: A semi-asynchronous hierarchical federated recommendation system in smart city," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 18 808–18 820, 2023.

[43] Z. Li, Z. Lin, F. Liang, W. Pan, Q. Yang, and Z. Ming, "Decentralized federated recommendation with privacy-aware structured client-level graph," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 4, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3641287

[44] Z. Li, G. Long, and T. Zhou, "Federated recommendation with additive personalization," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=xkXdE81mOK

[45] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, p. 225–331, mar 2009. [Online]. Available: https://doi.org/10.1561/1500000016

[46] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

[47] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BJ0hF1Z0b

[48] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008, pp. 111–125. [Online]. Available: https://doi.org/10.1109/SP.2008.33

[49] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753. [Online]. Available: https://doi.org/10.1109/SP.2019.00065

[50] NPR, "Facebook manipulates our moods for science and commerce: A roundup," 2014. [Online]. Available: https://www.npr.org/sections/alltechconsidered/2014/06/30/326929138/facebook-manipulates-our-moods-for-science-and-commerce-a-roundup

[51] H. Oosterhuis and M. de Rijke, "Differentiable unbiased online learning to rank," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1293–1302. [Online]. Available: https://doi.org/10.1145/3269206.3271686

[52] P. R. Ovi and A. Gangopadhyay, "A comprehensive study of gradient inversion attacks in federated learning and baseline defense strategies," in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, 2023, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CISS56502.2023.10089719

[53] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2429–2443. [Online]. Available: https://doi.org/10.1145/3548606.3560557

[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[55] V. Perifanis and P. S. Efraimidis, "Federated neural collaborative filtering," *Know.-Based Syst.*, vol. 242, no. C, apr 2022. [Online]. Available: https://doi.org/10.1016/j.knosys.2022.108441

[56] T. Qin and T. Liu, "Introducing LETOR 4.0 datasets," *CoRR*, vol. abs/1306.2597, 2013. [Online]. Available: http://arxiv.org/abs/1306.2597

[57] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 425–10 433. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.01044

[58] S. Ramaswamy, O. Thakkar, R. Mathews, G. Andrew, H. B. McMahan, and F. Beaufays, "Training production language models without memorizing user data," *CoRR*, vol. abs/2009.10031, 2020. [Online]. Available: https://arxiv.org/abs/2009.10031

[59] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. [Online]. Available: http://arxiv.org/abs/1707.04131

[60] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax," *Journal of Open Source Software*, vol. 5, no. 53, p. 2607, 2020. [Online]. Available: https://doi.org/10.21105/joss.02607

[61] J. Ren, X. Feng, B. Liu, X. Pan, Y. Fu, L. Mai, and Y. Yang, "Torchopt: An efficient library for differentiable optimization," *Journal of Machine Learning Research*, vol. 24, no. 367, pp. 1–14, 2023. [Online]. Available: http://jmlr.org/papers/v24/23-0191.html

[62] Reuters, "Amazon copied products and rigged search results to promote its own brands, documents show," 2021. [Online]. Available: https://www.reuters.com/investigates/special-report/amazon-india-rigging/

[63] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Techniques, applications, and challenges," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. New York, NY: Springer US, 2022, pp. 1–35. [Online]. Available: https://doi.org/10.1007/978-1-0716-2197-4_1

[64] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y

[65] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.05122

[66] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1703.03864

[67] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2017, pp. 3–18. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2017.41

[68] Z. Sun, Y. Xu, Y. Liu, W. He, L. Kong, F. Wu, Y. Jiang, and L. Cui, "A survey on federated recommendation systems." 2022. [Online]. Available: https://arxiv.org/abs/2301.00767

[69] Tamber, "Steam video games," 2017. [Online]. Available: https://www.kaggle.com/datasets/tamber/steam-video-games

[70] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–17, 2022. [Online]. Available: https://doi.org/10.1109%2Ftnnls.2022.3160699

[71] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 2815–2823. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00293

[72] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy preserving targeted advertising," in *Proceedings Network and Distributed System Symposium*, 2010. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/toub.pdf

[73] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[74] S. Wang, B. Liu, S. Zhuang, and G. Zuccon, "Effective and privacy-preserving federated online learning to rank," in *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ser. ICTIR '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3–12. [Online]. Available: https://doi.org/10.1145/3471158.3472236

[75] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[76] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. [Online]. Available: https://doi.org/10.1109/TIP.2003.819861

[77] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does llm safety training fail?" in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 80 079–80 110. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/fd6613131889a4b656206c50a8bd7790-Paper-Conference.pdf

[78] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020. [Online]. Available: https://doi.org/10.1109/TIFS.2020.2988575

[79] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft, "Blurme: Inferring and obfuscating user gender based on ratings," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, ser. RecSys '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 195–202. [Online]. Available: https://doi.org/10.1145/2365952.2365989

[80] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Communications*, vol. 13, no. 1, jun 2022. [Online]. Available: https://doi.org/10.1038%2Fs41467-022-30714-9

[81] L. Yang, B. Tan, V. W. Zheng, K. Chen, and Q. Yang, "Federated recommendation systems," in *Federated Learning: Privacy and Incentive*, Q. Yang, L. Fan, and H. Yu, Eds. Cham: Springer International Publishing, 2020, pp. 225–239. [Online]. Available: https://doi.org/10.1007/978-3-030-63076-8_16

[82] W. Yuan, C. Yang, Q. V. H. Nguyen, L. Cui, T. He, and H. Yin, "Interaction-level membership inference attack against federated recommender systems," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1053–1062. [Online]. Available: https://doi.org/10.1145/3543507.3583359

[83] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019. [Online]. Available: https://doi.org/10.1109/TNNLS.2018.2886017

[84] R. Zhang, S. Guo, J. Wang, X. Xie, and D. Tao, "A survey on gradient inversion: Attacks, defenses and future directions," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed. International Joint Conferences on

Artificial Intelligence Organization, 7 2022, pp. 5678–5685, survey Track. [Online]. Available: https://doi.org/10.24963/ijcai.2022/791

[85] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 3, apr 2020. [Online]. Available: https://doi.org/10.1145/3374217

[86] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf

APPENDIX A

A. Closed-form Solution for RAIFLE

We present below the derivation for a closed-form solution for $\mathcal{I}'$ under certain assumptions regarding the learning algorithm g.

We are given items $\mathcal{X}$, a model f with global model parameters θ, updated model parameters $\hat{\theta}$, and learning algorithm g. Suppose that $\mathcal{L}_{atk}$ is the L_2 loss and $\mathcal{L}_{FL}$ is the pointwise L_2. We further assume that:

$$\begin{aligned} g(\mathcal{X}, \theta, \mathcal{I}') &= \nabla_\theta \mathcal{L}_{FL}(\mathcal{I}', f(\mathcal{X}, \theta)) \\ &= \nabla_\theta ||\mathcal{I}' - f(\mathcal{X}, \theta)||_2^2 \\ &= -2\nabla_\theta f(\mathcal{X}, \theta) \cdot (\mathcal{I}' - f(\mathcal{X}, \theta)) \end{aligned} \tag{8}$$

Thus, we have:

$$\nabla_{\mathcal{I}'} g(\mathcal{X}, \theta, \mathcal{I}') = -2\nabla_\theta^T f(\mathcal{X}, \theta) \tag{9}$$

and consequently:

$$\nabla_{\mathcal{I}'}^2 g(\mathcal{X}, \theta, \mathcal{I}') = \mathbf{0} \tag{10}$$

Therefore, by theorem 1, there exists a global optimum for RAIFLE. Setting $\nabla_{\mathcal{I}'} \mathcal{L}_{atk}(\hat{\theta}, \theta')$ to $\mathbf{0}$ (from eq. 6), we have:

$$\begin{aligned} &\nabla_{\mathcal{I}'} \mathcal{L}_{atk}(\hat{\theta}, \theta') = \mathbf{0} \\ \iff &\nabla_{\mathcal{I}'} g(\mathcal{X}, \theta, \mathcal{I}') \cdot (\hat{\theta} - g(\mathcal{X}, \theta, \mathcal{I}')) = \mathbf{0} \\ \iff &\nabla_{\mathcal{I}'} g(\mathcal{X}, \theta, \mathcal{I}') \cdot \hat{\theta} = \nabla_{\mathcal{I}'} g(\mathcal{X}, \theta, \mathcal{I}') \cdot g(\mathcal{X}, \theta, \mathcal{I}') \end{aligned} \tag{11}$$

Substituting the results from eq. 8 and 9 into eq. 11, we can now derive the closed-form solution for $\mathcal{I}'$:

$$\begin{aligned} &\nabla_{\mathcal{I}'} \mathcal{L}_{atk}(\hat{\theta}, \theta') = \mathbf{0} \\ \Leftrightarrow &\nabla_\theta^T f(\mathcal{X}, \theta) \cdot \hat{\theta} = -2\nabla_\theta^T f(\mathcal{X}, \theta) \nabla_\theta f(\mathcal{X}, \theta) \cdot (\mathcal{I}' - f(\mathcal{X}, \theta)) \\ \Leftrightarrow &\mathcal{I}' = f(\mathcal{X}, \theta) - \frac{1}{2} \left(\nabla_\theta^T f(\mathcal{X}, \theta) \nabla_\theta f(\mathcal{X}, \theta)\right)^{-1} \nabla_\theta^T f(\mathcal{X}, \theta) \cdot \hat{\theta} \end{aligned} \tag{12}$$

Note that $\nabla_\theta^T f(\mathcal{X}, \theta) \nabla_\theta f(\mathcal{X}, \theta)$ is invertible iff $\mathbf{rank}(\nabla_\theta f(\mathcal{X}, \theta)) = m$. Therefore, eq. 12 is a valid closed-form solution if and only if $\mathbf{rank}(\nabla_\theta f(\mathcal{X}, \theta)) = m$.

B. Indirect Manipulation's Image Quality

To measure the similarity of the manipulated images to the originals in our federated OLTR experiment with image-based data (Section V-C), we calculated the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) [76]. Overall, the average SSIM ranges from 0.70 to 0.80 and the average PSNR ranges from 25 to 30 across all tested computer vision models (Table XIII). These values indicate that the adversarial images are relatively similar to the originals but not without visible artifacts or distortions. While our research is not focused on creating the most subtle manipulations, it is straightforward to include a regularization loss based on SSIM or PSNR in the manipulation process to improve image similarity, likely at the expense of attack performance. We leave this for future research.

TABLE XIII. SSIM AND PSNR OF MANIPULATED IMAGES AND ORIGINALS (ROUNDED TO 2ND DECIMAL PLACE)

Model	SSIM Mean	SSIM Std.	PSNR Mean	PSNR Std.
ResNet18	0.70	0.06	25.80	1.39
RegNet Y 800MF	0.80	0.06	29.36	1.27
DenseNet121	0.77	0.05	27.60	1.67
MNasNet 1.3	0.76	0.06	28.00	1.30

C. Discussion of Additional Countermeasures

We briefly discuss several hypothetical defense mechanisms against RAIFLE, including PIR, self-attacking, and alternative FL protocols. While these approaches are not yet as practical as the ones discussed in Section VII, they are nonetheless theoretically interesting to consider.

1) Private Information Retrieval: PIR [19], [40] aims to allow a user to retrieve an item without the server knowing the identity of the item. However, if a user participating in a PIR protocol also participates in IFL, then the privacy of their retrieved items can potentially be broken via our fingerprinting technique. Consider a user u and an item t: the server is interested in whether u has retrieved t or not. Similar to bypassing SA, the server can fingerprint a feature of t and zero out the values of that feature for all items other than t. Thus, if the local update from user u has a non-zero update to the feature parameters corresponding to the fingerprinted feature, it must be the case that t was included in u's training data. Once again, this demonstrates the need to apply DP noise as the fingerprint would no longer be guaranteed to be preserved.

2) Using RAIFLE as Guardrail: Although we devise RAIFLE to demonstrate the increased privacy risks in IFL, it could also be repurposed as a form of "sanity check" before users send out their local updates. Users would attack their own local updates and items to see if their true interactions can be reliably reconstructed or not. If significant leakage is observed, users can choose not to share the local updates or to rerun their defense mechanisms with stronger privacy guarantees (e.g. reduce ε in DP). We envision self-attacking as a red-teaming technique to empirically assess privacy leakage in IFL. This approach has two drawbacks: (1) users not being able to reconstruct their interactions does not necessarily mean the server would also be unable to, and (2) users will have to dedicate extra computation resources to run the attack.

To avoid a false sense of security, results should only be interpreted conservatively: success indicates likely leakage, while failure still leaves leakage potential.

3) Alternative FL Protocols: RAIFLE's threat model assumes a central FL server capable of controlling the interaction items. One natural possible defense is to adopt a different FL architecture or protocol with no single centralized authority, targeting the core assumption of our attack. For example, in the peer-to-peer gossip-based decentralized recommendation model [30], [43], users exchange model weights/gradients and/or item interactions with other peers in the network, thus eliminating the need for a central server. Hierarchical FL is another example where users are grouped into clusters and communicate exclusively with some statically or dynamically determined "leaders" of the clusters [75], [42]. While these scenarios can prevent RAIFLE thanks to their network topology, they often assume that peers are honest-but-curious and thus can still be vulnerable to Sybil attacks where malicious peers collude to gain an oversized influence on the FL network and uncover user interaction data [12]. Additionally, even though the peers are also assumed to be able to communicate anonymously with one another, in reality, practical anonymous communication protocols often incur additional networking overhead (on top of the already increased communication expense from FL decentralization) and can still be subject to various deanonymization attacks [36]. That said, we believe these "serverless" FL schemes hold great potential to further protect user privacy, and we hope to see more development and adoption from both academia and industry.

D. t-SNE Visualization of Gradients Under Manipulation

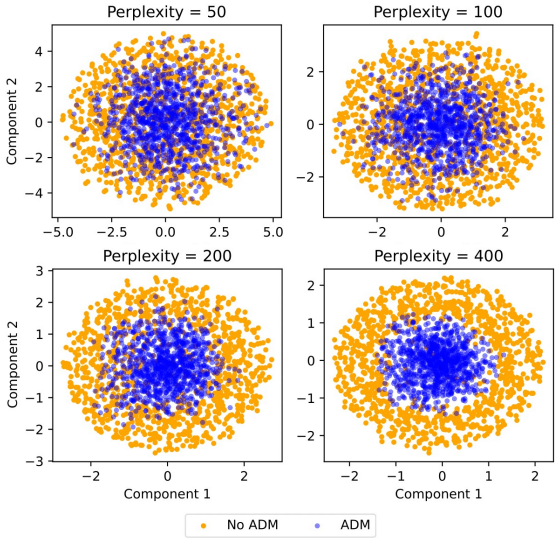

Fig. 8. t-SNE visualization of the local FL gradients for neural FPDGD with 16 hidden units on MSLR-WEB10K, with and without ADM.

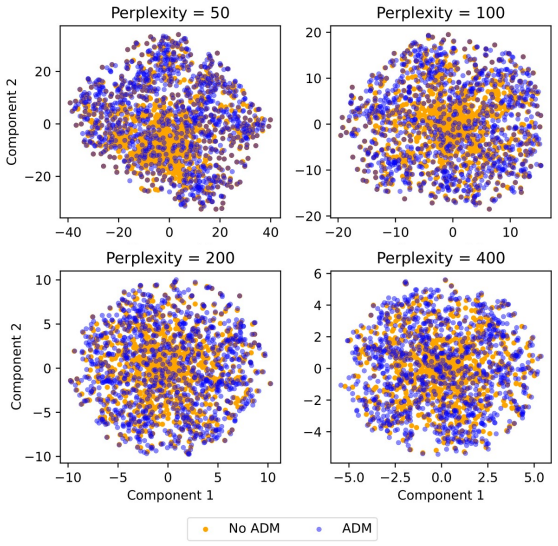

Fig. 9. t-SNE visualization of the local FL gradients for image-based FOLTR with ResNet18 and a neural ranker (8 hidden units), with and without ADM.

APPENDIX B
ARTIFACT APPENDIX

A. Description & Requirements

1) How to access:

- DOI: https://doi.org/10.17605/OSF.IO/HDSQR
- Github: https://github.com/dzungvpham/raifle

2) Hardware dependencies:

- A commodity machine with at least 16GB of RAM and 30GB of storage.
- Optional but highly recommended: A CUDA-capable NVIDIA GPU with at least 8GB of VRAM (preferably 12-16GB)

3) Software dependencies:

- A (x86-64) Unix-based OS. (Windows WSL will probably also work, but might need some installation modifications.)
- conda (such as Miniconda).

4) Benchmarks: We evaluate on the following datasets:

- Recommendation: MovieLens-100K, STEAM-200K
- Learning-to-rank: MQ2008, MQ2007, MSLR-WEB10K
- Image: ImageNet 2012 (validation split)

B. Artifact Installation & Configuration

We use conda to set up the environment. Please see the README file in our repository for detailed instructions on how to set up as well as download the relevant datasets.

C. Experiment Workflow

Please refer to the Evaluation section.

D. Major Claims

- (C1): RAIFLE achieves near-perfect AUC and significantly outperforms the state-of-the-art attack (IMIA) on federated recommendation (particularly the federated neural collaborative filtering algorithm), as demonstrated by experiment (E1) and reported in Tables V and VI of Section VI-A for MovieLens-100K and STEAM-200K.
- (C2): RAIFLE outperforms traditional gradient inversion on federated online learning-to-rank, as demonstrated by experiments (E2) and (E3) and reported in Table VII for MQ2007 and MSLR-WEB10K (Section VI-B) and Table VIII for ImageNet (Section VI-C).

E. Evaluation

All of our experiments are in IPython Jupyter Notebook inside the `code` folder. We recommend using Visual Studio Code to run our experiments. More detailed step-by-step instructions can be found in the README of our repository. For artifact evaluation, we make some suggested modifications to our code (e.g., omit some experiment configurations and reduce the number of simulations/users) to keep the evaluation under the time limit. Full-scale runs can be easily enabled by modifying our notebooks as instructed in the code comments.

Each experiment has the following structure:

- The central FL server initializes a global FL model, performs manipulation on the FL training item features, then shares the model and items with users
- Each user trains the global FL model locally using the manipulated data and their private interactions, applies differential privacy noise, then sends back the updated model weights to the server (directly or via secure aggregation).
- The server inverts the gradient updates to guess the private interactions.

1) Experiment (E1): Federated Recommendation Systems: We run RAIFLE on the MovieLens-100K and STEAM-200K datasets with the Federated Neural Collaborative Filtering (FNCF) algorithm.

[Execution] Run all cells in `experiment_rec.ipynb` in order. The default dataset is MovieLens-100K. Cell 2 contains instructions on how to change the dataset. For artifact evaluation, we scaled down the number of users attacked to 30, which will take about 30 minutes to finish.

[Results] After the experiment completes, the raw results are saved as `output/rec_metrics.csv` and a summary of the AUC and F1 score is printed out. The `name` column describes the configuration in the format: FNCF_eps_{epsilon}_IMIA_{reg_factor},

where {epsilon} refers to the local DP ε parameter (inf means no DP). IMIA_0.0 means the IMIA defense is not applied, IMIA_1.0 means the IMIA defense is applied with L1 regularization factor 1.0.

	auc							
	count	mean	std	min	25%	50%	75%	max
name								
FNCF_eps_1.0_IMIA_0.0	3.0	0.460413	0.051742	0.400720	0.444391	0.488062	0.490259	0.492456
FNCF_eps_1.0_IMIA_1.0	3.0	0.503129	0.024825	0.484654	0.489020	0.493387	0.512367	0.531348
FNCF_eps_100.0_IMIA_0.0	3.0	0.525789	0.050219	0.492867	0.496889	0.500911	0.542251	0.583591
FNCF_eps_100.0_IMIA_1.0	3.0	0.509239	0.073901	0.439126	0.470649	0.502173	0.544296	0.586420
FNCF_eps_20.0_IMIA_0.0	3.0	0.493490	0.029515	0.466049	0.477878	0.489707	0.507211	0.524714
FNCF_eps_20.0_IMIA_1.0	3.0	0.483418	0.086971	0.417729	0.434104	0.450479	0.516263	0.582047
FNCF_eps_500.0_IMIA_0.0	3.0	0.757722	0.145745	0.596186	0.696896	0.797607	0.838490	0.879372
FNCF_eps_500.0_IMIA_1.0	3.0	0.529289	0.042971	0.502927	0.504496	0.506066	0.542470	0.578875
FNCF_eps_inf_IMIA_0.0	3.0	0.998888	0.001175	0.997659	0.998333	0.999007	0.999503	1.000000
FNCF_eps_inf_IMIA_1.0	3.0	0.696291	0.140118	0.580646	0.618384	0.656121	0.754114	0.852107

Fig. 10. Example output of the federated RS experiment on MovieLens-100K for only 3 users

2) Experiment (E2): Federated Online Learning to Rank (FOLTR) (non-image): We run RAIFLE on the MQ2007, MQ2008, and MSLR-WEB10K datasets with the Federated Pairwise Differentiable Gradient Descent (FPDGD) algorithm.

[Execution] Run cell 1, 2, and 3 in `experiment_ltr.ipynb`. Cell 2 contains instructions on how to change the dataset and other configs. For artifact evaluation, the default configuration is a linear ranker + a neural net ranker with 16 hidden units, MQ2007 dataset, and 16 queries per user (this will take about 5 hours, a full run on MQ2007 can take more than 1 day, MSLR-WEB10K is much longer).

[Results] After the experiment completes, the raw results are saved as `output/ltr_metrics.csv` and a summary of the AUC is printed out. The `name` column describes the configuration in the format:

{model_name}_{click_model_name}_{num_query}_query_eps_{epsilon}_{key},

where {model_name} is 'linear_pdgd', 'neural_16_pdgd', etc., {click_model_name} is either 'informational' or 'navigational', {num_query} is the number of queries per user (e.g., 16), {epsilon} is the local DP epsilon (inf means no DP), and {key} is either 0.0 or 1.0, where 0.0 means no manipulation and 1.0 means full manipulation.

	auc							
	count	mean	std	min	25%	50%	75%	max
name								
linear_pdgd_informational_16_query_eps_1.0_0.0	3.0	0.546499	0.057720	0.508406	0.513294	0.518182	0.565545	0.612909
linear_pdgd_informational_16_query_eps_1.0_1.0	3.0	0.498567	0.087459	0.422609	0.450759	0.478909	0.536545	0.594182
linear_pdgd_informational_16_query_eps_100.0_0.0	3.0	0.569947	0.012843	0.560386	0.562648	0.564909	0.574727	0.584545
linear_pdgd_informational_16_query_eps_100.0_1.0	3.0	0.686625	0.070352	0.621091	0.649455	0.677818	0.719392	0.760966
linear_pdgd_informational_16_query_eps_20.0_0.0	3.0	0.559078	0.061512	0.489636	0.535253	0.580870	0.593798	0.606727
linear_pdgd_informational_16_query_eps_20.0_1.0	3.0	0.570062	0.041321	0.527636	0.550002	0.572367	0.591274	0.610182
linear_pdgd_informational_16_query_eps_500.0_0.0	3.0	0.567336	0.048591	0.511455	0.551186	0.590918	0.595277	0.599636
linear_pdgd_informational_16_query_eps_500.0_1.0	3.0	0.743431	0.041398	0.697273	0.726511	0.755749	0.766511	0.777273
linear_pdgd_informational_16_query_eps_inf_0.0	3.0	0.592894	0.033480	0.565411	0.574251	0.583091	0.606636	0.630182
linear_pdgd_informational_16_query_eps_inf_1.0	3.0	0.834709	0.049419	0.782000	0.812063	0.842126	0.861063	0.880000

Fig. 11. Example output of FOLTR experiment on MQ2007 for only 3 users with the default configuration.

3) Experiment (E3): FOLTR with ImageNet: We run RAIFLE on the ImageNet dataset with a simple pointwise ranking algorithm. This is the only experiment where a GPU is recommended.

[Execution] Run cell 1, 2, 3, and 4 in `experiment_ltr_cv.ipynb`. The default configuration (scaled down for artifact evaluation) is ResNet18 as feature extractor, 30 rounds of simulation, and 5,000 images. Cell 2 contains instructions on how to change the feature extractor. Cell 3 generates the manipulated images. The batch size may need to be adjusted depending on how much GPU memory is available, e.g., 128 if 8GB, 256 if 12GB or more. Cell 3 can take 2-3 hours with this configuration. If a GPU is not available, we suggest using `raifle_ltr_cv_colab.ipynb` with Google Colab's GPU runtime instead. Cell 4 simulates the attack and can take 2-3 hours (without GPU).

[Results] After the experiment completes, the raw results are saved as `output/ltr_cv_metrics.csv` and a summary of the AUC is printed out. The `name` column describes the configuration in the format: {model_name}_{num_items}_items_eps_{epsilon}_{key}, where {num_items} is 512, 1024, or 2048 (assuming ResNet18), {epsilon} is the local DP epsilon (inf means no DP), and {key} is 'no_adm' (no manipulation) or 'adm_opt' (RAIFLE).

	auc							
	count	mean	std	min	25%	50%	75%	max
name								
neural8_2048_items_eps_1.0_adm_opt	3.0	0.507547	0.018772	0.493053	0.496944	0.500835	0.514793	0.528751
neural8_2048_items_eps_1.0_no_adm	3.0	0.510524	0.011834	0.497440	0.505546	0.513652	0.517066	0.520480
neural8_2048_items_eps_100.0_adm_opt	3.0	0.524530	0.006972	0.516979	0.521434	0.525889	0.528306	0.530722
neural8_2048_items_eps_100.0_no_adm	3.0	0.528793	0.017625	0.508460	0.523324	0.538187	0.538959	0.539732
neural8_2048_items_eps_20.0_adm_opt	3.0	0.503775	0.009867	0.494875	0.498469	0.502063	0.508224	0.514386
neural8_2048_items_eps_20.0_no_adm	3.0	0.507843	0.016747	0.497890	0.498176	0.498461	0.512820	0.527178
neural8_2048_items_eps_500.0_adm_opt	3.0	0.564693	0.039491	0.520930	0.548282	0.575473	0.586574	0.597674
neural8_2048_items_eps_500.0_no_adm	3.0	0.549535	0.025451	0.522611	0.537763	0.552794	0.562997	0.573200
neural8_2048_items_eps_inf_adm_opt	3.0	0.985450	0.015063	0.968733	0.979192	0.989650	0.993809	0.997968
neural8_2048_items_eps_inf_no_adm	3.0	0.925811	0.038281	0.885688	0.907747	0.929807	0.945872	0.961937

Fig. 12. Example output of FOLTR experiment on ImageNet for only 3 simulation rounds with the default configuration.