Towards Fair Federated Recommendation Learning: Characterizing the Inter-Dependence of System and Data Heterogeneity

Kiwan Maeng
Meta
Cambridge, MA, USA
kwmaeng@fb.com

Haiyu Lu
Meta
Menlo Park, CA, USA
hylu@fb.com

Luca Melis
Meta
New York, NY, USA
lucamelis@fb.com

John Nguyen
Meta
Menlo Park, CA, USA
ngjhn@fb.com

Mike Rabbat
Meta
Montreal, Québec, Canada
mikerabbat@fb.com

Carole-Jean Wu
Meta
Cambridge, MA, USA
carolejeanwu@fb.com

ABSTRACT

Federated learning (FL) is an effective mechanism for data privacy in recommender systems that runs machine learning model training on-device. While prior FL optimizations tackled the data and system heterogeneity challenges, they assume the two are independent of each other. This fundamental assumption is *not* reflective of real-world, large-scale recommender systems — *data and system heterogeneity are tightly intertwined.* This paper takes a data-driven approach to show the inter-dependence of data and system heterogeneity in real-world data and quantifies its impact on the overall model quality and fairness. We design a framework, RF^2, to model the inter-dependence and evaluate its impact on state-of-the-art model optimization techniques for federated recommendation tasks. We demonstrate that the impact on fairness can be severe under realistic heterogeneity scenarios, by up to 15.8–41× compared to a simple setup assumed in most (if not all) prior work. The result shows that modeling realistic system-induced data heterogeneity is essential to achieving fair federated recommendation learning.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Modeling and simulation.*

ACM Reference Format:
Kiwan Maeng, Haiyu Lu, Luca Melis, John Nguyen, Mike Rabbat, and Carole-Jean Wu. 2022. Towards Fair Federated Recommendation Learning: Characterizing the Inter-Dependence of System and Data Heterogeneity. In *the Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3523227.3546759

RecSys '22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9278-5/22/09…$15.00
https://doi.org/10.1145/3523227.3546759

1 INTRODUCTION

Recommender systems are a fundamental building block of modern internet services, empowering day-to-day applications. They suggest videos on Netflix [17] and YouTube [15], musics on Spotify [33], apps on the Google Play Store [12], and stories on Instagram [50]. A recent study showed that 60% of YouTube's and 75% of Netflix's videos watched were selected based on recommender systems [14, 65, 73]. Recommendation systems are one of the important machine learning workloads, comprising 50% of the training [2] and 80% of inference cycles [21] at Meta in 2019.

While recommender systems were traditionally trained inside datacenters, recent studies are increasingly exploring training the models on client devices, using *federated learning* (FL) [53, 54]. FL is a privacy-enhancing training method that is already well-adopted in many commercial products for non-recommendation use-cases, including Google's Gboard [23, 75] and Meta's Oculus keyboard [22]. FL trains a model locally on each client device using its local data and later aggregates only the model updates. FL does not require raw user data to leave the client device.

Training models with FL faces several challenges due to the data and system heterogeneity of participating clients [35]. *Data heterogeneity* means data in each user device is not *independent and identically distributed* (IID), hampering convergence [35]. *System heterogeneity* means client devices (e.g., smartphones) have widely varying system capabilities, which limits the model capacity and training efficiency [35]. In particular, to tackle system heterogeneity, many prior works proposed various *tier-aware optimizations* [6, 10, 16, 28, 39, 42], which apply different levels of optimizations to each device tier based on the system capabilities (Section 2.2.3).

However, when studying the tier-aware optimizations, no prior work looked at the *inter-dependence* of data and system heterogeneity, assuming the two are independent of each other. Prior work used a random mapping approach to model data and system heterogeneity simultaneously [16, 28, 42, 74], which always produce zero correlation between the two (Section 3.2). By analyzing data from a large-scale recommender system deployment, we show that the simplistic assumption is not representative of the real world – in real systems, data and system heterogeneity are tightly intertwined (Section 3.2). We refer to the tight correlation as ***system-induced***

data heterogeneity. We show that the system-induced data heterogeneity in real data can cause optimizations to experience fairness issues, which is a phenomenon not observed in prior work. To the best of our knowledge, this is the first time system-induced data heterogeneity and its effects are demonstrated.

Based on this observation, we developed RF^2 (Realistic Federated Recommendation for Fairness), an FL framework for recommender systems that simulates system-induced data heterogeneity. RF^2 includes: (1) code to simulate FL using popular recommendation models and datasets, (2) a statistical method tocontrol system-induced data heterogeneity, and (3) implementations of popular FL optimizations for system heterogeneity [7, 16, 23, 28, 39]. Our evaluation with RF^2 reveals that popular FL optimizations can hurt the model fairness severely when realistic system-induced data heterogeneity is present, sometimes by more than 40× compared to a no system-induced data heterogeneity case. Our evaluation also lists several interesting observations. We show that methods that showed similar fairness implications with no system-induced data heterogeneity can show significantly different fairness impacts with realistic system-induced data heterogeneity. We also show optimizations that achieve the best accuracy are not always the fairest (e.g., two similar-accuracy optimizations can differ in their fairness by 4.88×). We hope our evaluation motivates the need to simulate more realistic system-induced data heterogeneity, which RF^2 achieves. Our key contributions are:

(1) We identify the existence of system-induced data heterogeneity and its potential effects in real-world data. To the best of our knowledge, this work is the first to explicitly reveal such effects in the real world.
(2) We propose a method to synthesize system-induced data heterogeneity onto existing datasets. Datasets generated with our method can simulate interesting fairness effects of the real world, while prior approaches cannot.
(3) We present RF^2, an FL simulation framework for recommendation models that can simulate system-induced data heterogeneity and various FL optimizations. RF^2 is open-sourced at https://github.com/facebookresearch/RF2.
(4) Our evaluation lists several effects of system-induced data heterogeneity on existing optimizations. We hope the findings will inspire future researchers to design and evaluate fair FL systems on a more realistic setup.

2 BACKGROUND AND MOTIVATION

2.1 Deep Learning Recommender Systems

Recommender systems suggest items to users by predicting the likelihood of an interaction (e.g., click or purchase) between a user and items. We broadly use the term *click* to refer to any positive user-item interaction. Various techniques have been explored to deliver high-quality recommendations, ranging from classical techniques, e.g., matrix factorization [40], to emerging new deep learning-based techniques [12, 20, 51, 68, 69, 78, 79], just to name a few. In this paper, we will focus on deep learning-based approaches and refer to them as recommender systems.

Deep learning-based recommender systems use features of users and items as inputs to predict whether a user will click a particular item. Two commonly-used feature types are dense features and sparse features. Dense features represent features of continuous values, such as a user's age or the price of an item. Sparse features represent categorical features of discrete values, such as a user's gender, the collection of items a user liked in the past, or the genre of a movie. Sparse features are usually encoded as an extremely sparse one- or multi-hot vector.

To predict the click probability, recommender systems first translate sparse features into dense embedding vectors using embedding tables [12, 51, 79]. The embedding vectors are merged with dense features and go through a multi-layer perception (MLP), producing a prediction at the end. Different model architectures explore variations in how the features are merged, including simple concatenation [25], element-wise multiplication [25], pairwise dot product [12, 51], attention-based weighted averaging [79], or using another deep model [69, 78].

2.2 Federated Learning

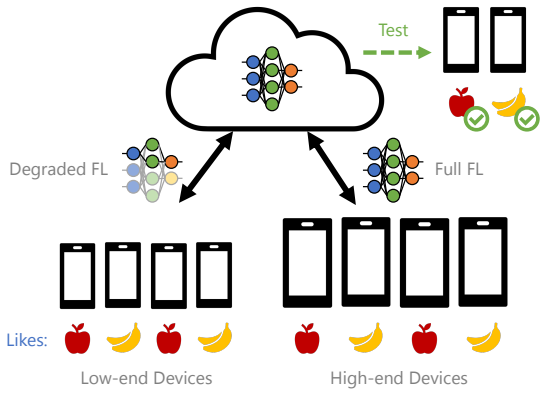
Federated learning (FL) [23] trains a model using a pool of client devices without each client having to send its data to the server. In this section, we discuss the workflow of FL and how prior literature handles data and system heterogeneity.

2.2.1 Workflow of Federated Learning. To train a model using FL, a centralized server first selects clients to participate from a client pool. The selected clients download the model from the server and train it locally using their data. After training, the clients upload their trained models (or equivalently, the gradients) back to the server. When all the participating clients upload their gradients, the server aggregates the gradients and updates the server-side model. The process repeats until the model converges. In the most commonly-used FedAvg algorithm [23], the server aggregates client gradients using weighted averaging, where the number of samples in each client corresponds to a weight value. Then, the aggregated gradient is simply added to the server model or applied using a separate server-side optimizer [66].

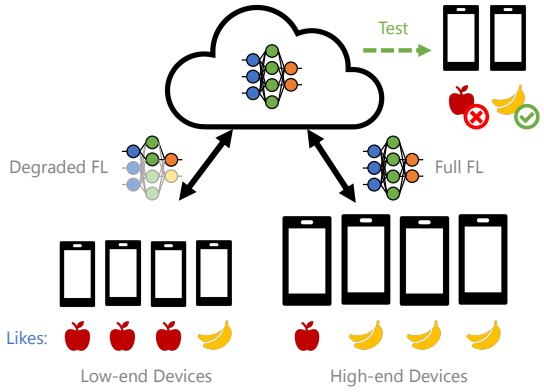
2.2.2 Data Heterogeneity. FL is a form of distributed ML training. However, unlike distributed training in datacenters where data can be shuffled so that each trainer node has an independent and identically distributed (IID) subsample [45], the data of each FL client is non-IID — the number of samples and the feature/label distributions on each client are different from each other [35]. Data heterogeneity makes it challenging to reach high model quality [30]. Many algorithms [1, 23, 37, 59, 67] have been proposed to improve the model quality in the presence of data heterogeneity.

2.2.3 System Heterogeneity and Tier-Aware Optimizations. Client devices (e.g., smartphones) vary significantly in their system capabilities, including computing power, memory, storage, and network speed [41, 71, 74]. For example, low-end and high-end smartphones may experience a 2–6× latency difference when training the same model [74] and two orders of magnitude difference in their network bandwidth [64]. The *system heterogeneity* degrades the efficiency of FL because each round in FL proceeds only after all the participating clients finish training. The synchronous nature makes slow clients become *stragglers* that bottleneck the entire training process.

To mitigate the straggler effect, recent studies proposed *tier-aware optimizations*. The core idea is to group devices with similar



(a) A case without system-induced data heterogeneity



(b) A case with system-induced data heterogeneity

Figure 1: Tier-aware optimizations can hurt fairness when system-induced data heterogeneity is present. The figure shows an optimization that makes low-end devices train only a subset of the model [10, 16, 28]. The optimization produces a fair model if the data distribution between low- and high-end devices are similar (a), but may become unfair if the data distribution differ significantly (b).

system capabilities into *tiers* and apply distinct optimizations to different device tiers, so that lower-tier devices bear lighter computation/communication burdens. Below, we describe some of the commonly proposed forms:

Excluding low-end devices. The simplest optimization is to prevent low-end devices from participating in FL entirely to minimize the presence of stragglers. This simple solution can either be implemented by explicitly leaving out low-end devices [23] or by implicitly setting a training time deadline that low-end devices cannot meet [7]. Many real products have adopted this strategy. For example, Google's Gboard's next-word prediction disallows devices with less than 2GB RAM from participating in FL [23].

Over-selection and dropping. Another well-adopted optimization is to select $N\%$ more clients than needed during selection and drop the slowest $N\%$ during aggregation [52]. Low-end devices are more likely to be dropped by this optimization because they are more likely to end up being the slowest $N\%$.

Tiered gradient compression. When there is a network bandwidth imbalance between tiers, applying gradient compression (e.g., gradient pruning [8, 39, 44, 77] or quantization [4, 39]) more aggressively to devices with a slower network can balance the communication speed. Not all techniques from other use-cases are applicable to FL, however. For example, the popular Top-K pruning [47] may leak which entries of the embedding tables were accessed in FL [53].

Tiered model sizes. When model computation time imbalance is severe, using smaller models for devices with less computing capabilities can relieve the imbalance. Several prior work proposed using a smaller number of channels for low-tier devices to reduce computation time and memory usage [10, 16, 28]. Upon model aggregation, channels are only averaged across tiers that use the channels [10, 16, 28], and knowledge distillation can be additionally used to further improve model accuracy [28]. Others allowed each device tier to use an entirely different model from each other and relied on knowledge distillation to aggregate the knowledge [11, 13, 24, 28, 34, 43, 46]. Figure 1a illustrates an example of a tier-aware optimization, where low-end devices train a smaller model with fewer channels in its hidden layers.

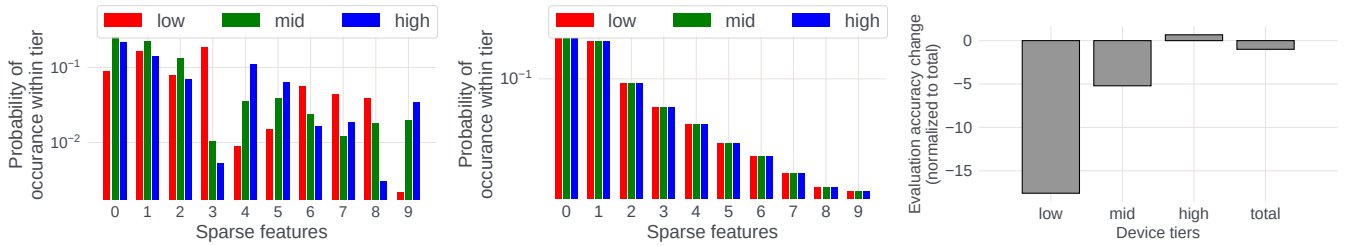
3 REAL-WORLD OBSERVATIONS: DATA AND SYSTEM HETEROGENEITY ARE INTERTWINED

3.1 Inter-dependence Between Data and System Heterogeneity

Section 2.2.2–2.2.3 discussed a stream of prior research that tackled the data heterogeneity and system heterogeneity of FL. However, most (if not all) prior studies tackled data and system heterogeneity separately, assuming *no inter-dependence exists between the two*. This assumption, however, is not reflective of the real world.

As a motivating example, assume that there are clients who like apples and clients who like bananas in the world, and their fruit preferences are an important feature of a recommender system (e.g., the system recommends apple juice to apple-liking clients). If the probability of liking apples or bananas is the same regardless of the client's device tier as in Figure 1a, we say there exists no inter-dependence between data and system heterogeneity.

Alternatively, there can be cases where the probability of liking apples is higher for low-end devices, while the probability of liking bananas is higher for high-end devices (Figure 1b). When there is such data distribution difference *between device tiers*, inter-dependence exists between data and system heterogeneity. We term such an inter-dependence as ***system-induced data heterogeneity*** in this work. When system-induced data heterogeneity exists, applying tier-aware optimizations may cause fairness issues. For example, if we use fewer channels for low-end devices, and low-end devices mostly hold apple-liking features, the final trained model may not work as well for apple-liking clients as for banana-liking clients, because most of the apple-liking data were trained through a model with fewer channels (Figure 1b). We show in Section 3.2 that real-world recommender systems experience system-induced data heterogeneity, and the fairness of the model can be impacted



(a) Real-world statistics **(b) Random-mapping statistics** **(c) Fairness implications**

Figure 2: Real world experiences system induced data heterogeneity that can impact fairness. (a) plots the distribution of feature values across tiers in the real world, and (b) plots what the distribution would look like instead with random tier mapping. Comparing the two clearly shows that the real world experiences system-induced data heterogeneity. (c) shows the accuracy change for each tier when excluding low-end devices from training, in the presence of system-induced data heterogeneity. Low-end devices are disproportionately penalized compared to the overall population.

when tier-aware optimizations are applied without careful considerations. Thus, it is important to simulate realistic system-induced data heterogeneity when evaluating FL optimizations.

Unfortunately, no prior FL literature assumed the existence of system-induced data heterogeneity to the best of our knowledge, and hence no prior optimizations were evaluated in the presence of realistic system-induced data heterogeneity. When simulating data and system heterogeneity, even the advanced FL simulators with real-world system traces [41, 74] synthesized or collected client data that show data heterogeneity and *randomly assigned* synthesized or collected system heterogeneity characteristics (i.e., device tiers) to each client [16, 28, 41, 74]. Such a random tier-client mapping always produces a dataset with no system-induced data heterogeneity, as in Figure 1a.

3.2 Does the Real World Experience System-induced Data Heterogeneity?

To understand whether system-induced data heterogeneity exists in the real world, we analyzed important sparse features of a recommender system that serves billions of users worldwide. Figure 2a presents the statistics of a sparse feature that is known to be important in delivering high-quality recommendations (e.g., fruit preferences in Figure 1). We group the user devices into three tiers (low-, mid-, and high-) based on their system capabilities and observe how frequently each value in the feature (e.g., apple, banana, orange, ...) occurs within each tier. Figure 2a plots the result for the top-10 most frequently observed values. In Figure 2b, we plot the statistics again, but this time, by mapping users to tiers randomly as in prior work [16, 28, 41, 42, 74] instead of using the actual tiers.

Comparing Figure 2a and Figure 2b, it is clear that *real-world deployment environment experiences notable system-induced data heterogeneity*. When using random tier mapping (Figure 2b), the probability of each sparse feature value occurring is the same across tiers. In other words, the affinity to apples/bananas is the same across device tiers (Figure 1a). However, real-world data (Figure 2a) exhibits high data heterogeneity across tiers, resembling the scenario in Figure 1b. For example, sparse feature value 3 is mostly observed only in the low-end device tier, resembling the preferences for apple in Figure 1b. Value 9, on the other hand, is mostly observed in the mid/high-end device tiers but very scarcely in the low-end device tier, resembling the preferences for bananas in Figure 1b.

We also demonstrate that popular tier-aware optimizations can introduce fairness degradation in the presence of system-induced data heterogeneity. We trained a recommendation model using real-world data similar to Figure 2a, while (1) using all devices' data, and (2) not using low-end devices' data. The second setup follows Google's Gboard FL optimization [23]. Figure 2c shows the resulting model accuracy change after excluding low-end devices' data for each tier, normalized by the overall average. Low-end devices get disproportionately affected, suffering from 17.6× more accuracy degradation than the average population. Figure 2c motivates the need to study tier-aware optimizations in a realistic system-induced data heterogeneity setup. If not, model quality for certain populations can be significantly degraded unintentionally.

4 STUDYING SYSTEM-INDUCED DATA HETEROGENEITY FOR RECOMMENDER SYSTEMS

RF^2 is an FL simulation framework for recommender models that enables agile modeling of system-induced data heterogeneity. RF^2 supports (1) efficient FL training for popular recommender models and datasets (Section 4.1), (2) synthesizing varying degrees of system-induced data heterogeneity onto existing datasets (Section 4.2), (3) a family of tier-aware optimization strategies from prior work (Section 4.3), and (4) fairness evaluation (Section 4.4) that can guide programmers to refine and test their optimizations. Figure 3 illustrates the design overview of RF^2.

4.1 Simulating FL for Recommender Systems

RF^2 supports FL simulation for state-of-the-art, commonly-used recommender models and datasets. While FL for deep recommender models has been studied in previous literature [53, 55], prior frameworks were either confined to simplistic models that take only user ID and item ID as inputs [55] or were built on proprietary datasets [53]. RF^2, on the other hand, is compatible with a large body of popular recommender models, by being built on top of DeepCTR-Torch [61], an open-source codebase that implements

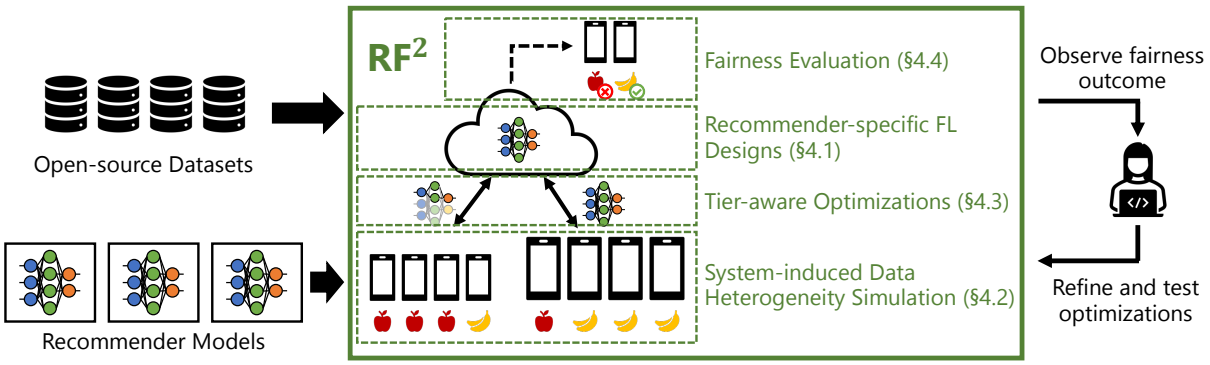


Figure 3: Overview of RF^2.

19 recommender models (in a non-FL context) and is easily extensible to more. RF^2 currently supports two commonly-used open-source datasets, Taobao Ad Display/Click Data [58] and MovieLens-20M [19] (Section 5.1), and can be extended to additional datasets.

RF^2 makes some unique design decisions to improve convergence and model a more realistic setup. Instead of using minibatch SGD on the client [53, 55], RF^2 implements an option to use a full-batch SGD. Full-batch SGD is practical because recommender systems tolerate a large batch size, [1] and clients usually do not have many datapoints as user-item interaction is rare. For example, the Taobao dataset [58] has only 26 datapoints on average per client. Using full-batch SGD on the clients and advanced optimizers, e.g., AdaGrad, on the server [66] improves the learning stability significantly. RF^2 does not select a client again before every client is selected exactly once, unlike prior work that models duplicated selection [9, 53, 55]. The non-duplicate selection is to simulate a more realistic large-scale FL, where billions of clients participate [7, 32] and duplicated selection is extremely rare.

Algorithm 1 Dirichlet-based tier mapping.

1: $seenUsers, tier_0, tier_1, tier_2 \leftarrow \emptyset, \emptyset, \emptyset, \emptyset$
2: **for** $item \in items.sortedByDescendingPopularity()$ **do**
3: $\quad p_0, p_1, p_2 \leftarrow Dirichlet(\alpha, 3)$
4: $\quad p_L, p_M, p_S \leftarrow sortDescending(p_0, p_1, p_2)$
5: $\quad tier_L, tier_M, tier_S \leftarrow sortByDescendingSize(tier_0, tier_1, tier_2)$
6: $\quad$ **for** $user \in item.clickedUsers()$ **do**
7: $\quad\quad$ **if** $user \notin seenUsers$ **then**
8: $\quad\quad\quad r \leftarrow random()$
9: $\quad\quad\quad$ **if** $r < p_L$ **then**
10: $\quad\quad\quad\quad tier_S \leftarrow tier_S \cup user$
11: $\quad\quad\quad$ **else if** $p_L \le r < p_L + p_M$ **then**
12: $\quad\quad\quad\quad tier_M \leftarrow tier_M \cup user$
13: $\quad\quad\quad$ **else**
14: $\quad\quad\quad\quad tier_L \leftarrow tier_L \cup user$
15: $\quad\quad\quad seenUsers \leftarrow seenUsers \cup user$

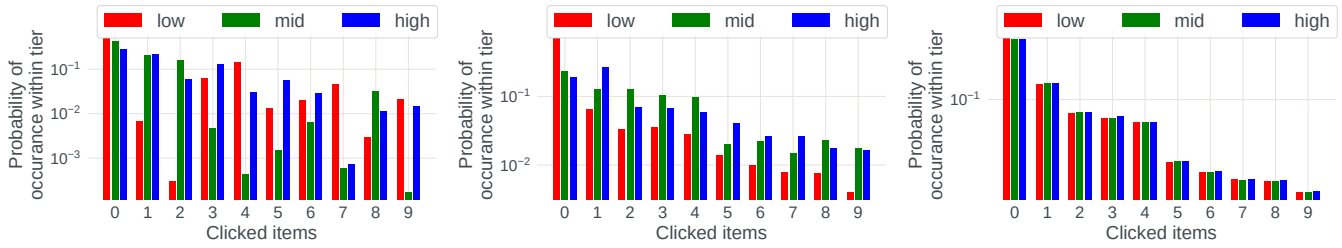
4.2 Simulating System-aware Data Heterogeneity

One of RF^2's main goals is to simulate realistic system-induced data heterogeneity. There are many potentially viable ways to simulate system-induced data heterogeneity. Across tiers, one can vary the distribution of user features, click rates, number of samples, or affinity to different items. We concentrate on making the *affinity to different items* heterogeneous across tiers (e.g., make certain tiers like certain items more, as in Figure 1b). Our approach is applicable to any recommender datasets as they always have click information that represents the user-item affinity.

Algorithm 1 shows how we assign tiers to each client to simulate system-induced data heterogeneity. Here, we assume three tier groups, $tier_0$, $tier_1$, and $tier_2$. Starting from the most popular item (Line 2), we draw three samples for the three tiers from a Dirichlet distribution [30] with a given α (Line 3). p_0, p_1, and p_2 represent the probability for each user who clicked this item to be in each tier. If α is small, the values are more skewed, leading to higher system-induced data heterogeneity. If α is high, system-induced data heterogeneity is reduced. We sort the three probabilities (Line 4) and also the number of already assigned users for each tier (Line 5), so that the tier with currently the smallest number of users ($tier_S$) gets the largest probability (p_L) of the user being assigned. Lines 4–5 ensure that the final number of users is similar across tiers, and can be omitted if balancing the number of users is undesired. With the given probability, each user that clicked the item (Line 6) gets assigned to one of the three tiers (Line 9–11), unless it is already assigned to a certain tier (Line 7). For users that never clicked any items, we treat them as clicking a null item and apply the same procedure. Our tier assignment procedure is inspired by the approach used to simulate data heterogeneity in FL across clients [30]. Our algorithm has a different goal, which is to simulate system-induced data heterogeneity (data heterogeneity *across tiers*).

Figure 4 shows the generated system-induced data heterogeneity using different α for the MovieLens-20M dataset [19] (see Section 5.1 for more details on the dataset). In the figure, the top-10 most clicked items and their occurrence on each (synthesized) tier are plotted. We can see that for low α (Figure 4a), the dataset experiences a severe system-induced data heterogeneity similar to that of the real world (Figure 2a). As we increase α (Figure 4b– 4c), the distribution becomes increasingly more similar to random mapping

[1] https://github.com/mlcommons/training_results_v1.1/blob/main/NVIDIA/benchmarks uses a batch size of 70k



(a) $\alpha = 0.5$ **(b)** $\alpha = 5$ **(c)** $\alpha = 5000$

Figure 4: Dirichlet distribution with different α can simulate different degrees of system-induced data heterogeneity.

(Figure 2b). The simulated system-induced data heterogeneity also leads to different fairness implications. Figure 5 shows an accuracy degradation of each tier when excluding low-end devices from FL under different system-induced data heterogeneity (see Sections 5 and 6 for details on the evaluation setup and results). Figure 5a shows that low-end devices experience disproportionate accuracy loss when there is high system-induced data heterogeneity, similar to what was observed in the real world (Figure 2c). The same level of fairness issue cannot be observed with low system-induced data heterogeneity (Figure 5b).

4.3 Supporting Popular Tier-Aware Optimizations in FL

RF^2 implements several commonly-used tier-aware optimizations discussed in Section 2.2.3. Specifically, the current version implements (1) excluding low-end devices (**Exclude Lo**), (2) overselection and drop (**Overselect**), (3) tier-aware gradient pruning (**Prune**), (4) tier-aware gradient compression (**Quant**), and (5) tier-aware channel width reduction (**Channel**). For simulation, a performance model for each tier was obtained from [74], which models the performance as a Gaussian distribution using real-world measurements. For pruning (**Prune**), we explored random pruning that is widely used in FL [39] and do not consider Top-K pruning [47] due to privacy concerns (Section 2.2.3). For quantization (**Quant**), we studied stochastic rounding [4, 39]. Given n bits, stochastic rounding uniformly splits the value range between the minimum and the maximum with 2^n uniformly separated points $p_0, ... p_{2^n-1}$. If $p_k < x < p_{k+1}$, then we round x into p_k with a probability of $\frac{x-p_k}{p_{k+1}-p_k}$ and p_{k+1} otherwise. We also explore a variant that uses 1 bit to encode the sign and $n-1$ bits to encode the absolute value (**QuantS**). The variant gives us a better representation of zero, which we will show to have fairness improvement in Section 6. To reduce computation, we focus on varying the channel dimensions (**Channel**) without additionally using knowledge distillation [16, 28]. Knowledge distillation-based approaches [11, 13, 24, 28, 34, 43, 46] require a representative public dataset which we do not assume. We reduced channels for all but the first hidden layer, as reducing channels for the first hidden layer effectively ignores some input features. RF^2 can be extended to support more optimizations.

4.4 Quantifying Fairness

To study whether an optimization strategy impacts each tier equally, we use the *relative accuracy change* [26, 27] for each tier before and after applying an optimization. Mathematically, if model accuracy is β_p^t for tier $t \in \{low, mid, high\}$ and an optimization p ($p = 0$ is no-optimization) is applied, the relative accuracy change for tier t is defined as $\frac{\beta_p^t - \beta_0^t}{\beta_0^t}$. To quantify the fairness impact of an optimization p across tiers, we report the *maximum difference in the accuracy change (**MDAC**)* between tiers. MDAC is higher if an optimization is more unfair, and 0 if perfectly fair. It is defined as:

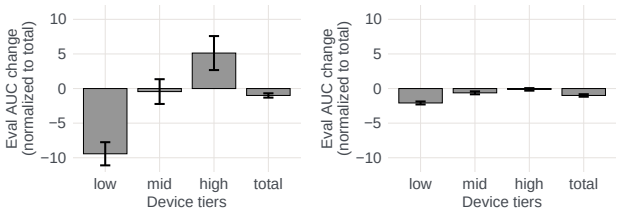
$$max(|\frac{\beta_p^{t_i} - \beta_0^{t_i}}{\beta_0^{t_i}} - \frac{\beta_p^{t_j} - \beta_0^{t_j}}{\beta_0^{t_j}}|), t_i, t_j \in \{low, mid, high\} \quad (1)$$

5 EVALUATION METHODOLOGY

5.1 Deep Learning Recommendation Models and Datasets

Datasets. We study two commonly-used open-source recommendation datasets, Taobao Ad Display/Click Data [58] (i.e., Taobao dataset) and MovieLens-20M [19] (i.e., MovieLens dataset). The Taobao dataset shows 26 million interactions (click/non-click) between 1.14 million users and 847 thousand item ads across an 8-day period. Each user has 9 sparse features (e.g., gender or occupation), each ad has one dense (price) and 5 sparse (e.g., category or brand) features, and each event has one sparse feature that encodes the "scenario" [58]. The MovieLens dataset provides 20 million movie ratings for 27 thousand movies from 138 thousand users, along with the genre information for each movie. To convert it into a click/non-click dataset, we considered a 5-star rating as click and others as non-click [79]. Following prior work [78, 79], we did not use user ID as a user feature for privacy. Instead, we augmented the user features with the user history of previously clicked items (ads, categories, and brands for Taobao, movies for MovieLens). For Taobao, we additionally used the day of the week information [79]. We applied logarithm to Taobao's item price feature because the range of the value is very large, from 0.01 to 100 million.

Models. We evaluated two state-of-the-art deep recommender models, DLRM [51, 72] and DIN [79]. We did not study models that directly use user IDs, e.g., NeuMF [25], for enhanced privacy. DLRM [51] is a model developed by Meta. In DLRM, dense features go through a bottom MLP and are mixed with the output of the



(a) $\alpha = 0.5$ **(b)** $\alpha = 5000$

Figure 5: Excluding low-end devices have a higher fairness impact with a higher system-induced data heterogeneity for the Taobao dataset.

embedding tables through a pairwise dot product. The output goes through a top MLP to produce the final prediction. DIN [79] is a model proposed by Alibaba. In DIN, the user history features go through an *attention layer* after the embedding tables, which predicts the importance of each history and gives a larger weight to the history that is more relevant to the current item. After being re-weighted, the features are concatenated with the dense features and go through an MLP for prediction. For both models, we used the top MLP with a single hidden layer of size 256, and embedding tables with a dimension of 16. For DLRM, we used a bottom MLP with a hidden layer size 16. For DIN's attention layer, we used two hidden layers of sizes 64 and 16 and used Dice activation [79] without batch normalization. For clients, we used full-batch SGD with lr=1.0 for both datasets. For the server, we used AdaGrad with lr=0.01 for Taobao and lr=0.1 for MovieLens.

We used ROC-AUC [31], or AUC for short, as the accuracy metric. AUC measures the model quality well when the labels are extremely biased (e.g., when most of the ads are not clicked) [36, 51, 78, 79]. As a reference, the achieved test AUC after 1 epoch of non-FL training was 0.6096/0.6049 for the Taobao dataset with DLRM/DIN and 0.7995/0.7666 for the MovieLens dataset with DLRM/DIN, being similar to prior work [79]. The achieved test AUC after FL training with all clients exactly once was 0.5966/0.5941 (Taobao, DLRM/DIN) and 0.7954/0.7538 (MovieLens, DLRM/DIN), which are the values used as a baseline AUC for our fairness metric (MDAC) calculation. It is hard to compare our FL results with prior work directly because no prior work trained the exact same datasets and models in an FL setup; however, the achieved AUC falls into a similar range as prior work that used similar datasets [53, 79].

5.2 Tier-Aware Optimizations

We explored six classes of tier-aware optimization techniques explained in Section 4.3 (Exclude Lo, Overselect, Prune, Quant, QuantS, Channel). For Prune, Quant, QuantS, and Channel, we explore three different configurations each, which impose roughly 1:2:4, 1:2:8, or 1:4:16 communication/computation overheads to low-, mid-, and high-end devices. Below list summarizes the 14 configurations we studied.

- **Exclude Lo** excludes low-end devices.
- **Overselect** selects and drops 20% extra clients.
- **Prune 1:2:4** prunes 75% (low), 50% (mid), and 0% (high) of the gradients.
- **Prune 1:2:8** prunes 87.5% (low), 75% (mid), and 0% (high) of the gradients.
- **Prune 1:4:16** prunes 93.75% (low), 75% (mid), and 0% (high) of the gradients.
- **Quant/QuantS 1:2:4** quantizes the gradients using 8 (low), 16 (mid), and 32bits (high).
- **Quant/QuantS 1:2:8** quantizes the gradients using 4 (low), 8 (mid), and 32bits (high).
- **Quant/QuantS 1:4:16** quantizes the gradients using 2 (low), 4 (mid), and 32bits (high).
- **Channel 1:2:4** uses 25% (low), 50% (mid), and 100% (high) of the original channel size.
- **Channel 1:2:8** uses 12.5% (low), 25% (mid), and 100% (high) of the original channel size.
- **Channel 1:4:16** uses 6.25% (low), 25% (mid), and 100% (high) of the original channel size.

5.3 System-Induced Data Heterogeneity

We evaluated the effect of varying levels of system-induced data heterogeneity by evaluating all the configurations on (1) random tier mapping (**Random**, no system-induced data heterogeneity), and (2) Dirichlet-based tier mapping using five different α: **Hetero-vlow** ($\alpha = 5000$), **Hetero-low** ($\alpha = 5$), **Hetero-mid** ($\alpha = 0.5$), **Hetero-high** ($\alpha = 0.05$), and **Hetero-vhigh** ($\alpha = 0.005$). The configurations represent very low to very high system-induced data heterogeneity.

6 EVALUATION RESULTS

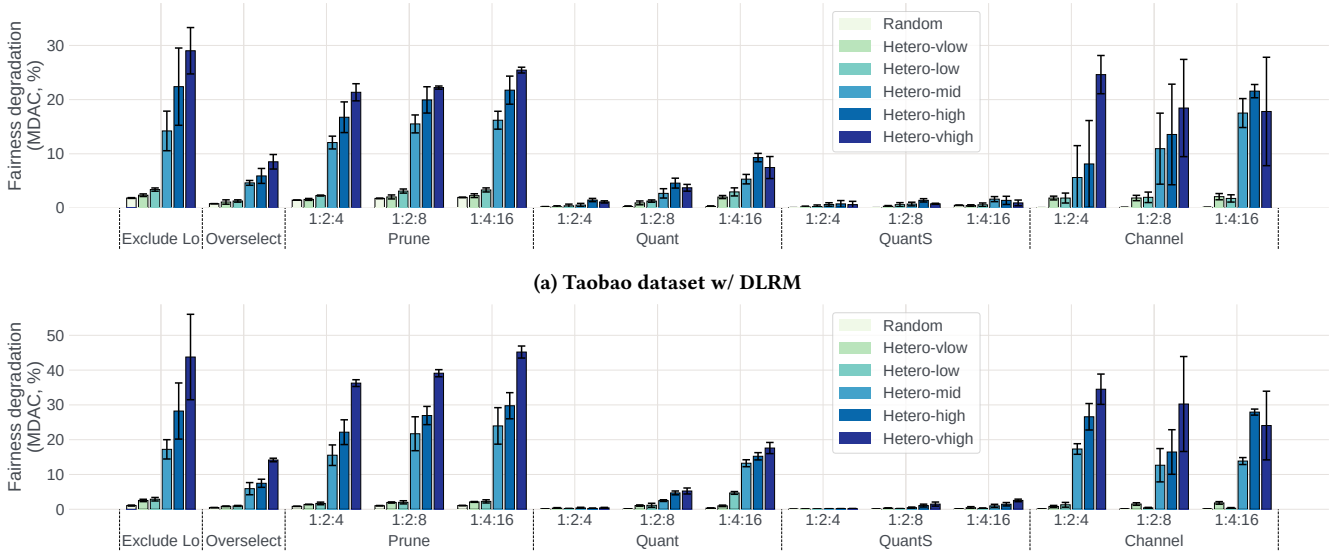
Our evaluation aims to answer the following questions in the presence of realistic system-induced data heterogeneity:

- How do tier-aware optimization strategies from prior literature affect fairness?
- How does the degree of system-induced data heterogeneity affect fairness?
- How do different models and datasets affect fairness?
- Is the best-performing optimization in terms of prediction accuracy also the best in terms of fairness?

6.1 Fairness Impacts of Different Optimizations Under System-Induced Data Heterogeneity

Figure 6–7 shows the results of training each model and dataset under the 14 optimization configurations and the 6 different system-induced data heterogeneity settings. The y-axis shows the fairness degradation (MDAC, defined in Section 4.4). A larger MDAC means the optimization strategy is more unfair.

Takeaway 1: Optimizations cause fairness degradation. In the presence of system-induced data heterogeneity (e.g., Hetero-vhigh/high), tier-aware optimizations may introduce significant fairness degradation. For example, Exclude Lo, which is an optimization used by Google [23], caused 29–44% MDAC with DLRM/DIN and Taobao dataset (Figure 6). The result means that low-end devices can suffer 29–44% more accuracy degradation than high-end devices in the presence of high system-induced data heterogeneity. Figure 6–7 also shows that more skewed tier-aware optimizations



(a) Taobao dataset w/ DLRM

(b) Taobao dataset w/ DIN

Figure 6: Different optimizations and different system-induced data heterogeneity have different fairness impacts. The figure plots the fairness implications of 14 different optimizations on 6 different heterogeneity levels on the Taobao dataset.

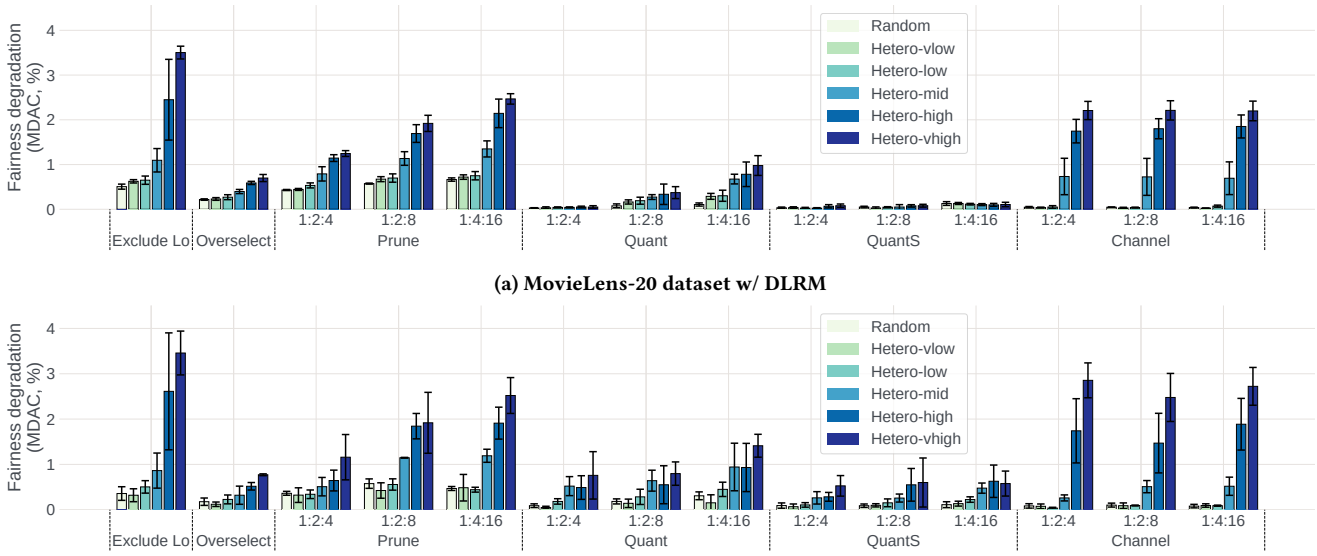
generally lead to a higher fairness degradation. For each optimization, the bar heights generally increase as we move from a less skewed optimization (1:2:4) to a more skewed optimization (1:4:16) in each optimization group.

Takeaway 2: Fairness impacts change depending on the degree of system-induced data heterogeneity. Figure 6–7 shows that more fairness is hampered when the degree of system-induced data heterogeneity is higher (Hetero-vhigh/high). Exclude Lo, for example, see more than 15.8× fairness degradation for DLRM/Taobao (MDAC 1.84% vs. 29%, Figure 6a), and 41× for DIN/Taobao (MDAC 1.06% vs. 43.7%, Figure 6b). The results imply that when studying tier-aware optimizations, simulating realistic system-induced data heterogeneity is crucial; otherwise, one might downplay the fairness implication of an optimization by up to 41×.

Takeaway 3: Different optimizations have different fairness impacts. Figure 6–7 also shows that some optimizations are fairer than the others in the presence of system-induced data heterogeneity. Take a look at Figure 6a, for example. By only looking at random mapping (Random), it may seem like Channel 1:2:4 brings similar fairness concerns with QuantS 1:2:8 (MDAC 0.045% vs. 0.044%). However, in the presence of system-induced data heterogeneity, QuantS 1:2:8 is much more fair than Channel 1:2:4 (MDAC 0.78% versus 24.62% for Hetero-vhigh, 1.41% versus 8.13% for Hetero-high, 0.73% versus 5.63% for Hetero-mid). This result again warns that only looking at random or low system-induced data heterogeneity cases might send a misguided message when assessing the fairness of optimizations. Among the methods we studied, Exclude Lo had the most unfair impact, while Quant/QuantS was the fairest.

Takeaway 4: Fairness impacts depend on the dataset/model architecture. Comparing Figure 6 and Figure 7, we can see that fairness also depends significantly on the characteristics of the dataset itself. The fairness impact of the optimizations is an order of magnitude larger for Taobao, compared to MovieLens (average MDAC 6.67% vs. 0.64% for DLRM + Hetero-vhigh, 6.45% vs. 0.55% for DLRM + Hetero-high). One hypothesis is that the rating of a movie is universal and easier to predict (i.e., a good movie is considered good by everybody) compared to Ads-clicks and, therefore, can be learned better even under a high degree of system-induced data heterogeneity. Similarly, comparing Figure 6 and Figure 7 reveals that DIN experiences slightly higher fairness degradation compared to DLRM (e.g., average MDAC 6.67% vs. 8.96% for DLRM + Hetero-vhigh). We can conclude that the fairness impact of different tier-aware optimizations heavily depends on both datasets and model architectures.

Takeaway 5: Quantization with separate sign encoding improves fairness. Comparing Quant with QuantS shows that QuantS impacts fairness much less. When comparing across all the configurations for high system-induced data heterogeneity scenarios (Hetero-vhigh/high/mid), QuantS 1:2:4 improves the fairness by 1.4 – 1.7× compared to Quant 1:2:4, QuantS 1:2:8 by 2.8–4× compared to Quant 1:2:8, and QuantS 1:4:16 by 4.9–5.9× compared to Quant 1:4:16. The reason is that while optimizations like pruning only lose gradient information within the tier if applied to a certain tier, quantization actually introduces *noise* in the gradient that can affect the model quality of other tiers. Particularly for embedding tables, gradients for the table entries that were not accessed by a certain client must be close to zero for that client. However, quantization may make these gradients non-zero as it will round zero into a nearby quantized value, introducing noise to un-accessed embedding entries. Because quantization with a separate sign encoding can better encode zero, it shows significantly better fairness results. The finding demonstrates a scenario where researchers can evaluate the fairness implications of their optimization proposals and modify their optimizations using RF^2.



(a) MovieLens-20 dataset w/ DLRM

(b) MovieLens-20 dataset w/ DIN

Figure 7: Different optimizations and different system-induced data heterogeneity have different fairness impacts. The figure plots the fairness implications of 14 different optimizations on 6 different heterogeneity levels on the MovieLens dataset.

6.2 Zooming into Each Optimization's Impact on Each Tier

Figure 8 illustrates the AUC change for each tier separately for one representative configuration — training DLRM with the Taobao dataset under Hetero-high. The results for the other setups showed similar trends and were omitted for space reasons. Zooming into the effect on each tier separately highlights additional observations.

Takeaway 6: Quantization benefits low-end devices, while all other optimizations punish low-end devices. As expected, most of the tier-aware optimization strategies degrade the model accuracy of the low-end devices disproportionately, because optimizations are more aggressively applied to resource-constrained, low-end devices. However, quantization degrades the model accuracy of mid/high-end devices more. The reason for the unexpected suffering of mid/high-end devices is again because quantized gradients of the low-end devices pollute the model updates of mid/high-end devices, especially from the embedding tables.

Takeaway 7: The best-accuracy optimization is not always the best-fairness optimization. When comparing the overall AUC degradation (the **total** bar group in Figure 8) with the fairness impact of each optimization (Figure 6), we can see that the optimizations that lead to minimal overall AUC degradation do not always coincide with optimizations that are the fairest. For example, Exclude Lo, which is one of the most unfair optimizations, shows reasonable AUC degradation (-1.26%) that is better than Quant 1:2:8 (-2.64%) and Quant 1:4:16 (-5.79%). However, Quant 1:2:8 and Quant 1:4:16 are much fairer (MDAC 4.59% and 9.3%, Figure 6a) than Exclude Lo (MDAC 22.4%, Figure 6a). This result indicates that only evaluating the overall accuracy after applying an optimization, as in the prior work [16, 28], may present an incomplete picture. Both the model accuracy and per-tier fairness (MDAC) must be considered to understand the overall design and optimization space better.

Overall, the key insights shared in this paper demonstrate that RF^2 can improve the fairness of real-world FL recommender systems by allowing optimizations to be tested under a more realistic system-induced data heterogeneity. Using RF^2, FL system designers can correctly understand the potential fairness implications of each tier-aware FL optimization and correctly choose or properly redesign optimizations that meet their accuracy/fairness goals.

7 ADDITIONAL RELATED WORK

Fairness in ML. Remotely related, many studies showed that applying optimizations on a trained model can disproportionately harm minorities in the dataset [26, 27]. A recent public study also showed that using smartphone data to train ML models can produce a model unfair towards groups without smartphones [3]. Our work shows how applying tier-aware optimizations during FL can impact groups with low-tier devices, studying distinguished aspects from these studies. Whether prior debiasing solutions [48, 60, 70] can be applied to our setup is an interesting future work.

FL simulation frameworks. Several simulation frameworks exist for FL [5, 9, 18, 41, 49, 56, 57, 74, 80]. Unlike RF^2, none of the prior simulators that we are aware of support simulating system-induced data heterogeneity, even the frameworks that focus on realistic system-heterogeneity simulation [41, 74]. These other simulators can adopt the core idea of RF^2's system-induced data heterogeneity simulation and implement it in their framework.

Other FL optimizations for system heterogeneity. In addition to the tier-aware optimizations we discuss in Section 2.2.3, other work proposed complementary solutions to tackle the system heterogeneity problem in FL. AutoFL [38] and OORT [42] use ML-based

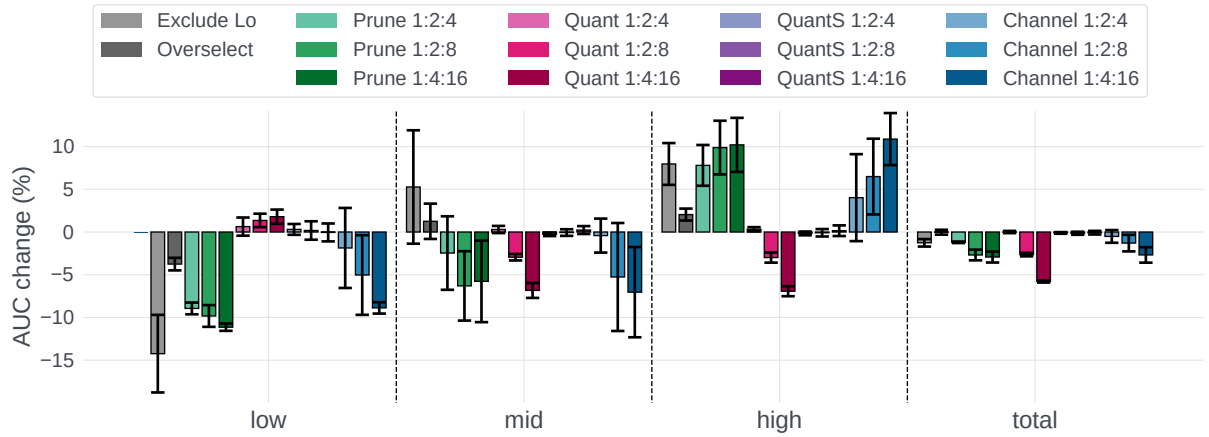


Figure 8: Different tiers are impacted differently for each optimization. The figure shows the case of training the Taobao dataset with DLRM with Hetero-high ($alpha = 0.05$) in more detail, plotting the AUC change for each tier separately.

client selection taking into account clients' system heterogeneity. FedBuff [52] and Papaya [32] implement asynchronous FL to mitigate the stragglers' effect. FedBuff still down-weighs slower clients' updates [52]. These systems were evaluated assuming no data-system inter-dependence, and studying the effect of system-induced data heterogeneity for these proposals will be an interesting future work.

Memory-efficient recommender systems. Training recommender systems on-device requires the models to be memory-efficient. Reducing MLP layers can be done by reducing the channel dimension [10, 16, 28], which was studied in this paper. Additionally, techniques were proposed to reduce memory usage of embedding tables on the server-side [29, 36, 62, 76]. Recent work [63] also proposed reconstructing embedding layers on-device during FL to reduce the memory footprint. These techniques have not been evaluated in the presence of system-induced data heterogeneity.

8 CONCLUSION

To enhance data privacy in recommender systems, federated learning has emerged as an effective mechanism. Despite a plethora of prior works on FL, an important characteristic of the real-world environment has not yet been considered. In this work, we shed light on the under-explored aspect of the inter-dependence between system and data heterogeneity — that has been considered individually but not in conjunction by most (if not all) prior work in the FL space. Based on the statistical observations from the real-world environment, we design a new statistical framework to model and evaluate the impact of system-induced data heterogeneity for federated recommendation learning. Our evaluation demonstrates that fairness can be severely affected under realistic system-induced data heterogeneity, and modeling the inter-dependence is essential to understanding the true fairness impacts.

ACKNOWLEDGMENTS

We would like to thank Kamalika Chaudhuri and Edward Suh for the invaluable discussion regarding the paper's direction. We thank Ilias Leontiadis, Shripad Gade, Mani Malek, Fangzhou Xu, Vlad Grytsun, and Shuaiwen Wang for their help in conducting the experiments. We also thank Ashkan Yousefpour, Sayan Ghosh, Hongyuan Zhan, Kaikai Wang, Dzmitry Huba, and Meisam Hejazi nia, who helped us understand the operation of federated learning system. We thank Pegah T. Afshar, Milan Shen, and Kim Hazelwood for supporting the work.

REFERENCES

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. 2021. Federated Learning Based on Dynamic Regularization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=B7v4QMR6Z9w

[2] Bilge Acun, Matthew Murphy, Xiaodong Wang, Jade Nie, Carole-Jean Wu, and Kim Hazelwood. 2021. Understanding Training Efficiency of Deep Learning Recommendation Models at Scale. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*.

[3] Alexis Stephens. 2014. Big Data Has Potential to Both Hurt and Help Disadvantaged Communities. https://nextcity.org/urbanist-news/big-data-good-bad-help-disadvantaged-communities.

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in Neural Information Processing Systems* 30 (2017).

[5] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).

[6] Ilai Bistritz, Ariana Mann, and Nicholas Bambos. 2020. Distributed distillation for on-device learning. *Advances in Neural Information Processing Systems* 33 (2020), 22593–22604.

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems* 1 (2019), 374–388.

[8] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. 2020. Adaptive Federated Dropout: Improving Communication Efficiency and Generalization for Federated Learning. *CoRR* abs/2011.04050 (2020). arXiv:2011.04050 https://arxiv.org/abs/2011.04050

[9] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[10] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210* (2018).

[11] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. 2019. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279* (2019).

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[13] Yae Jee Cho, Jianyu Wang, Tarun Chiruvolu, and Gauri Joshi. 2021. Personalized Federated Learning for Heterogeneous Clients with Clustered Knowledge Transfer. *arXiv preprint arXiv:2109.08119* (2021).

[14] Michael Chui, James Manyika, Mehdi Miremadi, N Henke, R Chung, P Nel, and S Malhotra. 2018. Notes from the AI frontier: Insights from hundreds of use cases. *McKinsey Global Institute* (2018).

[15] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[16] Enmao Diao, Jie Ding, and Vahid Tarokh. 2021. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=TNkPBBYFkXg

[17] Carlos A Gomez-Uribe and Neil Hunt. 2015. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)* 6, 4 (2015), 1–19.

[18] Google. 2022. TensorFlow Federated: Machine Learning on Decentralized Data. https://www.tensorflow.org/federated.

[19] Grouplens. 2016. MovieLens 20M Dataset. https://grouplens.org/datasets/movielens/20m/

[20] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[21] Udit Gupta, Carole-Jean Wu, Xiaodong Wang, Maxim Naumov, Brandon Reagen, David Brooks, Bradford Cottel, Kim Hazelwood, Mark Hempstead, Bill Jia, et al. 2020. The architectural implications of facebook's DNN-based personalized recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 488–501.

[22] Ian Hamilton. 2021. Oculus Quest Keyboard Option Sends 'Aggregate Modeling Data' To Facebook. https://uploadvr.com/facebook-quest-keyboard-data/

[23] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[24] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems* 33 (2020), 14068–14080.

[25] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[26] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. 2019. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248* (2019).

[27] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058* (2020).

[28] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. 2021. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems* 34 (2021).

[29] Oleksii Hrinchuk, Valentin Khrulkov, Leyla Mirvakhabova, Elena Orlova, and Ivan Oseledets. 2020. Tensorized embedding layers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 4847–4860.

[30] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).

[31] Jin Huang and Charles X Ling. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering* 17, 3 (2005), 299–310.

[32] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. 2022. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems* 4 (2022).

[33] Kurt Jacobson, Vidhya Murali, Edward Newett, Brian Whitman, and Romain Yon. 2016. Music personalization at Spotify. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 373–373.

[34] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2018. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479* (2018).

[35] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[36] Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H Chi. 2020. Learning to embed categorical features without embedding tables for recommendation. *arXiv preprint arXiv:2010.10784* (2020).

[37] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*. PMLR, 5132–5143.

[38] Young Geun Kim and Carole-Jean Wu. 2021. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 183–198.

[39] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[40] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[41] Fan Lai, Yinwei Dai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. FedScale: Benchmarking model and system performance of federated learning. In *Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning*. 1–3.

[42] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*. 19–35.

[43] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).

[44] Liang Li, Dian Shi, Ronghui Hou, Hui Li, Miao Pan, and Zhu Han. 2021. To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.

[45] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. 583–598.

[46] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 2351–2363.

[47] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887* (2017).

[48] Dugang Liu, Pengxiang Cheng, Hong Zhu, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2021. Mitigating Confounding Bias in Recommendation via Information Bottleneck. In *Fifteenth ACM Conference on Recommender Systems*. 351–360.

[49] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, et al. 2020. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987* (2020).

[50] Ivan Medvedev, Haotian Wu, and Taylor Gordon. 2019. Powered by AI: Instagram's Explore recommender system. https://ai.facebook.com/blog/powered-by-ai-instagrams-explore-recommender-system/.

[51] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).

[52] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Michael Rabbat, Mani Malek, and Dzmitry Huba. 2021. Federated learning with buffered asynchronous aggregation. *arXiv preprint arXiv:2106.06639* (2021).

[53] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2020. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[54] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. 2021. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503* (2021).

[55] Vasileios Perifanis and Pavlos S Efraimidis. 2022. Federated Neural Collaborative Filtering. *Knowledge-Based Systems* 242 (2022), 108441.

[56] Facebook Research. 2022. Federated Learning Simulator (FLSim). https://github.com/facebookresearch/FLSim.

[57] Jae Hun Ro, Ananda Theertha Suresh, and Ke Wu. 2021. FedJAX: Federated learning simulation with JAX. *arXiv preprint arXiv:2108.02117* (2021).

[58] Pavan Sabnagapati. 2020. Ad Display/Click Data on Taobao.com. https://www.kaggle.com/datasets/pavansanagapati/ad-displayclick-data-on-taobaocom

[59] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. 2018. On the convergence of federated optimization in

heterogeneous networks. *arXiv preprint arXiv:1812.06127* 3 (2018), 3.
[60] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 501–509.
[61] shenweichen. 2022. DeepCTR-Torch. https://github.com/shenweichen/DeepCTR-Torch
[62] Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 165–175.
[63] Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, John Rush, and Sushant Prakash. 2021. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems* 34 (2021).
[64] SpeedTest. 2022. Global Median Speeds March 2022. https://www.speedtest.net/global-index.
[65] C Underwood. 2019. Use cases of recommendation systems in business–current applications and methods.
[66] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. 2021. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917* (2021).
[67] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems* 33 (2020), 7611–7623.
[68] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.
[69] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*. 1785–1797.
[70] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1791–1800.
[71] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, Tommer Leyvand, Hao Lu, Yang Lu, Lin Qiao, Brandon Reagen, Joe Spisak, Fei Sun, Andrew Tulloch, Peter Vajda, Xiaodong Wang, Yanghan Wang, Bram Wasti, Yiming Wu, Ran Xian, Sungjoo Yoo, and Peizhao Zhang. 2019. Machine Learning at Facebook: Understanding Inference at the Edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.
[72] Carole-Jean Wu, Robin Burke, Ed H. Chi, Joseph Konstan, Julian McAuley, Yves Raimond, and Hao Zhang. 2020. Developing a Recommendation Benchmark for MLPerf Training and Inference.
[73] X Xie, J Lian, Z Liu, X Wang, F Wu, H Wang, and Z Chen. 2018. Personalized recommendation systems: Five hot research topics you must know. *Microsoft Research Lab-Asia* (2018).
[74] Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. 2021. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In *Proceedings of the Web Conference 2021*. 935–946.
[75] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).
[76] Chunxing Yin, Bilge Acun, Carole-Jean Wu, and Xing Liu. 2021. Tt-rec: Tensor train compression for deep learning recommendation models. *Proceedings of Machine Learning and Systems* 3 (2021), 448–462.
[77] Sixing Yu, Phuong Nguyen, Ali Anwar, and Ali Jannesari. 2021. Adaptive dynamic pruning for non-iid federated learning. *arXiv preprint arXiv:2106.06921* (2021).
[78] Wei Zhang, Wei Wei, Lingjie Xu, Lingling Jin, and Cheng Li. 2019. AI Matrix: A Deep Learning Benchmark for Alibaba Data Centers. *arXiv preprint arXiv:1909.10562* (2019).
[79] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.
[80] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. 2021. Pysyft: A library for easy federated learning. In *Federated Learning Systems*. Springer, 111–139.