# Understanding and Controlling Deanonymization in Federated Learning

Tribhuvanesh Orekondy[1] Seong Joon Oh[1†] Yang Zhang[2] Bernt Schiele[1] Mario Fritz[2]

[1] *Max Planck Institute for Informatics* [2] *CISPA Helmholtz Center for Information Security*
*Saarland Informatics Campus, Germany*

## Abstract

Federated Learning (FL) systems are gaining popularity as a solution to training Machine Learning (ML) models from large-scale user data collected on personal devices (e.g., smartphones) without their raw data leaving the device. At the core of FL is a network of anonymous user devices sharing training information (model parameter updates) computed locally on personal data. However, the type and degree to which user-specific information is encoded in the model updates is poorly understood. In this paper, we identify model updates encode subtle variations in which users capture and generate data. The variations provide a strong statistical signal, allowing an adversary to effectively deanonymize participating devices using a limited set of auxiliary data. We analyze resulting deanonymization attacks on diverse tasks on real-world (anonymized) user-generated data across a range of closed- and open-world scenarios. We study various strategies to mitigate the risks of deanonymization. As random perturbation methods do not offer convincing operating points, we propose data-augmentation strategies which introduces adversarial biases in device data and thereby, offer substantial protection against deanonymization threats with little effect on utility.

## 1 Introduction

Advances in machine learning (ML) is increasingly fueled by accessibility to data sources capturing rich representations of the world e.g., 9M photographs [49], 1.6M tweets [30], etc. While such large-scale data advances learning fundamental ML models (e.g., visual object recognition), the representations also encode a massive amount of unnecessary individual-specific information (e.g., person identities) [33, 61]. For situations where the data is decentralized (e.g., user-generated photos on edge devices), Federated Learning [54] provides a solution based on the principles of data minimization [43, 60] towards training a ML model. The core idea is participants distill from raw private data residing on individuals' device

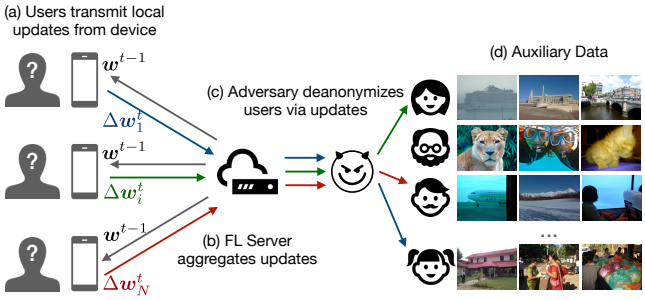



**Figure 1: Deanonymization in Federated Learning.** In this paper, we study how subtle user-biases captured in model parameter updates leads to deanonymization of their devices.

the information necessary to train the model, and intermittently communicate them to a server. The information communicated by the participants take the form of model updates computed locally on-device.

To prevent privacy violations, it is crucial that model updates reveals information solely necessary for the training task (e.g., visual features to identify cats) and nothing about the participants (e.g., person identities). To ensure this, federated learning is combined with additional steps to restrict the amount of data- and participant-specific information revealed in the process. In the specific case of restricting participant-specific information encoded in model updates, typical steps include: stripping the data of PII information [83], de-identifying the updates and auxiliary metadata [35, 54, 83], and avoiding authentication via user-identity prior to participation [16]. Hence, it is assumed that model updates received by the server contains minimal non-identifiable information to improve the model.

However, it is in the nature of many real-world federated settings, that the clients represent diverse users with different interests, preferences and habits. Hence, the underlying data distributions of the users are not identically distributed and as a consequence, is characteristic of the users. Therefore, we find that the model updates nonetheless en-

[†]SJO is currently at Clova AI Research, Naver Corp.

code individual-specific information and introduce *significant* deanonymization risks. Apart from constituting a privacy violation, deanonymization in federated learning undermines existing mechanisms to ensure the source of model updates are masked. Furthermore, deanonymization amplifies effectiveness of recent inference attacks (e.g., attribute inference [56]), as identities can be tied to sensitive attributes inferred from the participants' private training data.

We investigate deanonymization risks and consequences by following the popular Federated Averaging algorithm [15,54], where participating devices intermittently communicate de-identified model parameter updates to a server. Here, the high-dimensional updates are a product of multiple gradient steps on multiple batches of the local device data. We assume honest-but-curious server who intends to deanonymize participating devices (Fig. 1c) with limited access to prior information of users (Fig. 1d). Central to our deanonymization attack is exploiting subtle, but inherent, individual- specific biases introduced when participants collect data on personal devices. For instance, Alice capturing more photos of automobiles on her mobile device compared to Bob, who photographs food. Our approach learns a suitable representation where the biases (modeled from limited prior data) can be leveraged to re-identify individuals via their model updates.

We evaluate deanonymization risks in a federated learning setup when training complex models (e.g., MobileNet CNNs [44]) involving numerous participants (53-327 users). Furthermore, we use real-world (anonymized) user-generated datasets (e.g., PIPA, Blog) to closely emulate existing federated learning applications [54, 55]. Our evaluation indicates that participants can be consistently deanonymized across a range of scenarios. For instance, individuals transmitting model updates for an image classifier (with output classes e.g., chair, umbrella) on PIPA dataset are re-identified with high accuracy (19-175$\times$ chance-level). Furthermore, we find the attacks surprisingly possible in spite of a range of data-limited scenarios, such as when the adversary has only a single prior example of the targeted individual.

Moreover, we propose a novel cross-modal attack which tackles a challenging scenario when the attacker's prior information varies in modality from the private data used during training by the participants. For instance, the attacker leverages text information, while the participants are training using image data. Our experiments indicate that in spite of the cross-modal challenge, attacks are quite effective (0.76 AUC).

It is worth noting that our deanonymization attack can also amplify the performance of recent attacks that infer sensitive properties of the training data. For example, we show that learning an attack model to jointly perform deanonymization and attribute inference [56] are synergistic, with a consistent improvement of up to 4% accuracy on both tasks. These results are further concerning, as sensitive attributes can be linked to identities of participants in federated learning.

After demonstrating the the risks of deanonymization in federated learning, we explore countermeasures to mitigate the threat. We propose augmenting users' data distribution with an adversarial bias to decouple users' subtle variations from their prior information. As a result, we propose the first mitigation strategy that directly operates on the user data itself, while maintaining utility of the task. We find our strategy mitigate attacks with up to 95% effectiveness and incurs only negligible cost on the underlying task performance. In contrast, we find perturbation- and DP-based training approaches (e.g., DP-FedAvg [55]) incur large privacy and utility costs in our setup as they are typically effective only when training with a massive number of users (in the order of thousands).

## 2 Related Work

In this section, we position our paper with existing literature on anonymization and privacy in ML.

**Deanonymizing (Insufficiently) Anonymized Data**. Organizations have largely believed that explicitly stripping away key identification information (e.g., names, SSN) from data records is sufficient to de-identify and provide anonymity of participating individuals. Instances of this strategy to anonymize databases include Hospital Discharge dataset (GIC) [75], Netflix prize dataset [10], and AOL search logs [4]. However, a long of line work, dating back to [74, 75] highlight that although the de-identified database by itself might seem anonymous, joining with auxiliary publicly available data on a set of *quasi-identifiers* (e.g., zip-code, gender) leads to effectively re-identifying individuals. Consequently, in the aforementioned instances, many identities of participating individuals were re-identified using a public voter database [75], IMDb movie ratings [57], and search keywords [8] respectively. This has motivated significant research in the area of identifying factors that potentially lead to deanonymization of individuals, such as in social networks [58], programmatic code [2], and product reviews [39]. Research has also identified various sources of quasi-identifiers in unstructured data: profile attributes [65], geo-location [66], social graph structure [48], content [31], stylometric features [3], or RNA expressions [6]. In this paper, we tackle deanonymization of devices within Federated Learning, which enables users to anonymously participate towards the learning of an ML model using their private data.

**Attacks against Machine Learning Models**. Advances in ML has led to state-of-the-art statistical models being deployed 'in the wild' to perform a variety of tasks such as autonomous driving, fraud detection, and medical diagnosis. Attacks against such ML models can be targeted towards compromising the *integrity* of the model (such as by evasion attacks [5,13,18,32,63,76,79,84,87]), or its *privacy and confidentiality*. Our focus is on the latter, since ML models need to obviously learn something as a result of training on (potentially private and confidential) data sources. Attacks that

compromise privacy of models in this setting include: model stealing [52, 62, 77], membership inference [66, 67, 71] which identifies if a particular example was used during training, attribute inference [28, 56] to identify properties that holds true for subsets of data, and model inversion [27, 41] to reconstruct training class exemplars. In this work, we address a problem similar to membership and attribute inference, where we wish to identify properties that holds for subsets of data. While membership inference intends to identify whether a particular *example* was used during training, our adversarial goal can be cast as 'userbase inference': to identify which particular *individual* participated in training.

**Attacks in Federated Learning**. Distributed ML on decentralized private user-generated data sources – also referred to as Collaborative ML [70], or Federated Learning [15, 54] – is gaining popularity as it securely enables large-scale ML on private data sources. While such approaches minimizes the privacy risks by keeping user in control of the raw private data, understanding the extent of privacy risks is gaining traction in the research community. Understanding these risks in this setting is crucial since FL is designed to learn from private data spanning hundreds to tens of thousands of users. Unfortunately, research in this area is minimal and work has only recently started to quantify these risks.

Given the considerable complexity of FL systems exposing many attack surfaces, we specifically focus on the anonymous *gradient information* communicated by devices to the server. One line of work studies malicious devices who exploit anonymity and secure aggregation protocols to mount poisoning and backdoor attacks [7, 12, 82] on the system. Orthogonal to this line of work, are attacks [56, 59] where the server is modeled as an adversary instead of the device. Since the server requires raw unencrypted access to gradient signals for aggregation, it opens up threats to mount inference attacks that violate users' privacy. Recently, [59] comprehensively explored membership inference on gradient parameters, including an analysis in an FL setting. While our work explores a similar idea – membership on a user-level – we aim to determine it without access to the *exact* training example(s) belonging to the user. A closely related work to our paper is [56], who propose an *attribute inference attack* i.e., using the aggregated gradient signal to infer certain sensitive attributes (e.g., gender, race) that is not significantly correlated with the main task trained by participating users (e.g., sentiment analysis, gender classification). In this work, we show that deanonymization complements and amplifies such attribute inferences, by enabling an adversary to additionally associate the sensitive attributes to an individual.

## 3 Background, Notation and Terminology

In this section, we provide the preliminaries to Federated Learning, within which we explore our threat model in the next section. At this point, we remark that research towards a Federated Learning system encompasses among many other things, architecture [15], optimization techniques [46, 54], strategies to improve communication [47], aggregation [16], implementation [1], and applications [21, 35, 83]. To keep the background in this section concise, we present key concepts to understand: (i) how devices generate model parameter updates using the `FederatedAveraging` [55] algorithm; and (ii) how users anonymously communicate the parameter updates to the server in FL [15, 56, 59].

**Notation and Learning Objective**. In supervised learning, the overall objective is to learn a mapping $f_{\boldsymbol{w}} : \mathcal{X} \to \mathcal{Y}$ of a model $f$ parameterized by $\boldsymbol{w} \in \mathbb{R}$. The idea is to learn the parameters which minimizes the empirical risk represented by a loss function $L$ on a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} H(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i} L(f_{\boldsymbol{w}}(\boldsymbol{x}_i),\ \boldsymbol{y}_i) \qquad (1)$$

In FL, data is partitioned across multiple devices $k \in \mathbb{K}$: $\mathcal{D} = \bigcup_k \mathcal{D}_k$. Using $H_k(\boldsymbol{w})$ to denote the objective solved locally on device $k$, the objective in Equation 1 can now be re-written as:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{k=1}^{K} \frac{n_k}{n} H_k(\boldsymbol{w}) \qquad (2)$$

**Federated Averaging Algorithm**. Given the data $\mathcal{D}_k$ partitioned among devices $k \in \mathbb{K}$, the objective is to learn parameters $\boldsymbol{w}$ of the model $f_{\boldsymbol{w}}$, in the presence of a server $S$. We use the popular `FederatedAveraging` algorithm [53, 54] (Algorithm 1) proposed specifically to perform training on non-IID and imbalanced decentralized data; this has also served as the footing for multiple prior works [16, 29, 55, 72]. The idea here is that training occurs over multiple rounds, where in each round $t$, a fraction of devices $k \in \mathbb{K}_t$ train models $f_{\boldsymbol{w}}$ using the local data $\mathcal{D}_k^{\text{private}}$ and only communicate incremental model update $\Delta \boldsymbol{w}_k^t$ towards the server's global model $\boldsymbol{w}^t$. The server aggregates (such as by averaging) parameter updates from multiple devices and shares back an updated improved model after each round. Over multiple rounds of communications, the devices converge to model parameters $\boldsymbol{w}^T$ that has been effectively learnt from all the data $\mathcal{D}$, without their raw data ever being communicated to the server or another device. It should be noted that although we consider the simple `FederatedAveraging` algorithm, we expect our results to generalize to a broad class of decentralized algorithms which involve periodically exchanging model parameter updates.

**De-identification in FL**. A number of precautions are employed to ensure any identifiable information is stripped away from per-device update reports (which includes parameter updates $\Delta \boldsymbol{w}_k^t$ and additional metadata). We first iterate over de-identification strategies employed on-device. The client is initially registered into the FL process by being assigned pop-

**Algorithm 1:** `FederatedAveraging` [54] for training data on multiple devices

**Server's algorithm:**

**Input:** $K$ devices; $T$ number of rounds; $C$ fraction of devices sampled each round; $B$ device's batch size; $E$ number of local epochs

Randomly initialize $w^{t=0}$

**for** *round* $t \leftarrow 1$ **to** $T$ **do**

  $M \leftarrow \max(1, C \cdot K)$

  $\mathbb{K}_t \leftarrow$ sample $M$ devices from $\mathbb{K}$

  **for** *client* $k \in \mathbb{K}_t$ **do**

    $\Delta \boldsymbol{w}_k^{t+1} \leftarrow \texttt{DeviceUpdate}(k, \boldsymbol{w}^t)$

  **end**

  $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \sum_{k \in \mathbb{K}_t} \frac{n_k}{n} \Delta \boldsymbol{w}_k^{t+1}$

**end**

$\texttt{DeviceUpdate}(k, \boldsymbol{w}^t)$ :

$\mathcal{B} \leftarrow$ split local data $\mathcal{D}_k^{\text{private}}$ into batches of size $B$

$\boldsymbol{w} \leftarrow \boldsymbol{w}^t$

**for** *local epoch* $i \leftarrow 1$ **to** $E$ **do**

  **for** *batch* $b \in \mathcal{B}$ **do**

    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla L(f_{\boldsymbol{w}}; b)$

  **end**

**end**

$\Delta \boldsymbol{w} \leftarrow \boldsymbol{w}^t - \boldsymbol{w}$

**return** $\Delta \boldsymbol{w}$

ulation identifier [83] and thereby bypassing the need to authenticate with a device or user identity [15]. When possible, PII information is stripped away from the training data [35] prior to training on-device. After a number of local training steps, the parameter updates $\Delta \boldsymbol{w}_k^t$ along with anonymized operational metrics [83] is transmitted by the device. A (trusted) shuffler [14] can be additionally employed to ensure the transmitted per-device update reports are further sanitized before reaching the server. The shuffler typically strips away a range of user-specific metadata (e.g., IP addresses, routing details) and batches the reports (reordering updates to disassociate timing ordering information). On the whole, multiple mechanism are in-place to ensure that only the essence of the update-reports (i.e., the parameter updates $\Delta \boldsymbol{w}_k^t$) are received by the server to aggregate updates. Consequently, for the rest of the paper, we assume access to *only* the parameter updates to perform deanonymization.

# 4 Deanonymization Attacks in Federated Learning

In this section, we begin by presenting our threat model to deanonymize devices. We then discuss an insight to why this threat arises and work towards our attack models.

## 4.1 Threat Model

To highlight deanonymization risks in Federated Learning [54], we analyze a scenario with $K$ honest users ($K \geq 2$) who collaboratively train an ML model $f_{\boldsymbol{w}} : \mathcal{X} \rightarrow \mathcal{Y}$ over multiple rounds. A server $S$ co-ordinates the training, by periodically collecting model updates from a random subset of users. The model update communicated by each user is a result of performing multiple gradient steps over multiple batches on their local private data (see `DeviceUpdate(.)` in Algo. 1). Furthermore, the model updates are stripped of identifiable metadata [16, 35, 83] (e.g., device identifiers) and are optionally shuffled [14] to obscure the source of each individual update. Prior to summarizing information from multiple updates, we assume the server observes only the essence of per-user model update (i.e., parameter updates $\Delta \boldsymbol{w}_{\text{anon}}^t$) to improve $f_{\boldsymbol{w}}$.

We investigate deanonymization through the lens of an honest-but-curious server (the 'adversary') during the training process who uses the model update as an attack surface. The inference-time objective of the adversary is to deanonymize the model update i.e., re-identify the user $u$ who generated $\Delta \boldsymbol{w}_{\text{anon}}^t$. Such a deanonymization objective undermines sanitization mechanisms which de-identify model updates, such as decoupling the update from user identity [15], stripping away identifiable metadata [35, 83], and blind-shuffling mechanisms [14]. Furthermore, deanonymization also serves as a stepping stone for amplifying information recovered from other inference attacks. For instance, as we show later in §6.1.3, deanonymization can be coupled with attribute inference attacks to improve attack performances and further associate recovered attributes with identities.

To deanonymize, the adversary leverages limited prior knowledge of users. Formally, our threat model performs:

$$f^{\text{adv}} : \Delta \boldsymbol{w}_{\text{anon}}^t \times \mathcal{D}_u^{\text{prior}} \rightarrow u \stackrel{?}{=} \text{anon} \tag{3}$$

Here, $\Delta \boldsymbol{w}_{\text{anon}}^t$ is the deanonymization target, which is a result of an anonymous user taking multiple gradient steps on her local data $\mathcal{D}_{\text{anon}}^{\text{private}}$. The adversary's auxiliary knowledge of users is denoted by $\{\mathcal{D}_u^{\text{prior}} : u \in \mathbb{U}\}$. We assume $\mathcal{D}_u^{\text{prior}}$ represents a limited set of data generated by user $u$ and is distinct from their private data i.e., $\mathcal{D}^{\text{prior}} \cap \mathcal{D}_u^{\text{private}} = \emptyset \; \forall u \in \mathbb{U}$. For instance, historical data collected by the service, or content publicly shared by the users. In Section 5.2, we further elaborate on how we model the adversary's prior knowledge, as it plays a significant role in deanonymization attacks.

## 4.2 Selection Bias and Biased Estimators

The core idea of our threat model is to use users' selection bias as an identification cue, which we hypothesize (and shortly verify) is consistent among both the users' prior data (known to adversary) and private device data (unknown to adversary). This implicit user selection biases arise from behavioral factors [11, 26, 38] that results in subtle variations of how humans

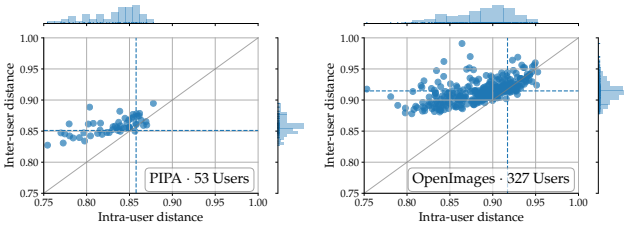



**Figure 2: Variations in user data.** Each point represents distances computed over the image set of a single user.

capture data. For instance, Alice's interest in automobiles might result in more variations of cars captured in her text/photos, compared to Bob whose interest lies in sports. At this point, we remark that this results in a non-IID data distribution among data on users and devices, which is well-known in FL literature [15,54]. However, we do identify and exploit the property that although the data is non-IID among users (large inter-user distances), the data displays lesser variation *within* data generated by the same user (small intra-user distances).

To validate the assumption, we now present an experiment to quantify user variations on two public image datasets (PIPA [86] and OpenImages [49]). In both cases, we (i) group the images based on the real-world user who captured them using the corresponding `author` fields; and (ii) vectorize images by extracting the 1024-dim `avgpool` features from MobileNet CNN [44] and $L_2$-normalize them. We obtain statistics for each user by computing two $L_2$ distances: (a) intra-user distance: median image feature distance between images within each user; and (b) inter-user distance: median image distance between user images and a set of random images. We plot these distances per user on a scatter plot in Figure 2, each point indicating a distinct user. If images captured by the users were unbiased, we would have found their corresponding points at the intersection of blue dashed lines. However, points predominantly being above the diagonal indicates that examples within each users' collection are similar (low intra-user distances), but are greater (high inter-user distances) when compared to other user collections. In Section 6.2.4, we further analyze how similar user-specific variations also arise in the parameter delta space.

The resulting non-IID distribution of user data $\mathcal{D}_u$ among devices leads to each device fitting a *biased* estimator during the `DeviceUpdate` step (Algo. 1) with a bias error: $\text{Bias}[\boldsymbol{w}_u] = \mathbb{E}[\boldsymbol{w}_u] - \boldsymbol{w}^*$, where the expectation term is over the user's training data $\mathcal{D}_u$ and $\boldsymbol{w}^*$ is the optimal estimator. We conjecture (validated in §6.2.4) that the bias error signal is consistently encoded in both: (i) the parameter updates transmitted by user's device $\Delta\boldsymbol{w}_u^t$; and (ii) when estimating on prior data of the user $\boldsymbol{w}_u^{\text{prior}} = \text{SGD}(\mathcal{D}_u^{\text{prior}})$. Hence, we reformulate the threat model (Eq. 3) in the parameter update space:

$$f^{\text{adv}} : \Delta\boldsymbol{w}_u^{\text{prior}} \times \Delta\boldsymbol{w}_{\text{anon}}^t \rightarrow u \stackrel{?}{=} \text{anon} \tag{4}$$

Next, we look at attack models to learn this mapping.

## 4.3 Attacks

In this section, we present attack models to deanonymize users based on their model updates (Eq. 4).

**Re-identification Attack.** In the re-identification scenario, the adversary leverages prior data to learn before-hand (via attack model $f^{\text{re-id}}$) what updates from targeted users look like. The adversary then uses the attack model to re-identify users based on their anonymous update. Formally, the re-identification attack involves training an attack model $f^{\text{re-id}} : \Delta\boldsymbol{w}_u^{\text{prior}} \rightarrow u$ to capture user-specific bias signals in the high-dimensional parameter delta space. At test-time, users are re-identified using their model updates:

$$f^{\text{re-id}} : \Delta\boldsymbol{w}_{\text{anon}} \rightarrow u \tag{5}$$

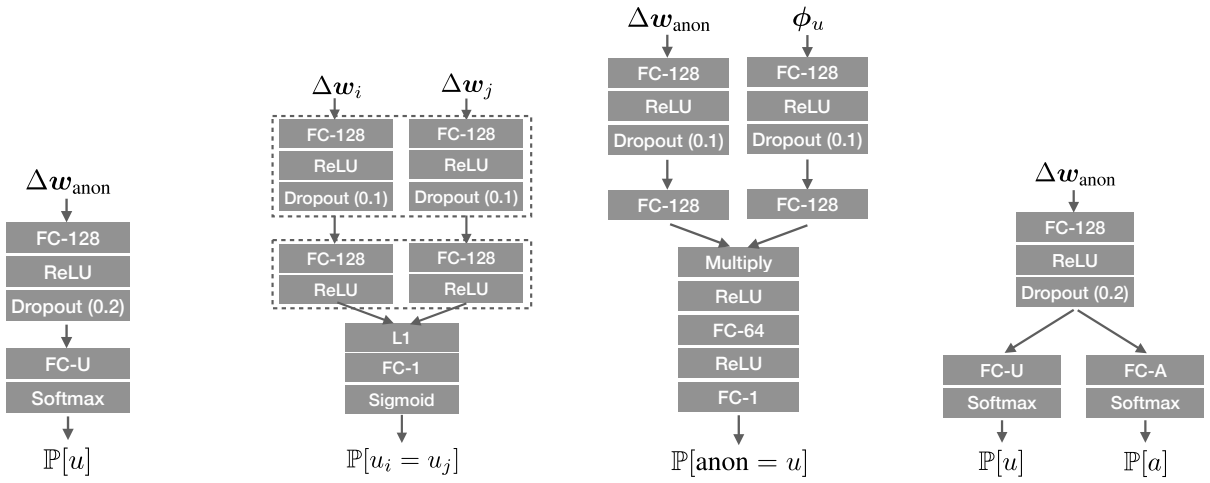
For the re-identification attack model $f^{\text{re-id}}$, we adopt a Multilayer Perceptron (MLP) classifier (architecture in Fig. 3a) with a single hidden layer of 128 units and ReLU activation, trained using SGD with learning rate (LR) 0.01, 0.9 momentum and $10^{-6}$ LR decay.

**Matching Attack.** Instead of learning an update-to-user mapping, the adversary in the matching scenario learns a metric space among model updates. Learning a metric space helps embed model updates close together if they are generated by the same user, independent of whether the user is a part of the adversary's prior knowledge base. Formally, the adversary's objective is to predict the match probability of a pair of distinct parameter updates:

$$f^{\text{mat}} : (\Delta\boldsymbol{w}_i,\ \Delta\boldsymbol{w}_j) \rightarrow i \stackrel{?}{=} j \tag{6}$$

where one or both parameter updates are anonymous. The matching attack is particular helpful in scenarios where the adversary encounters novel users at test-time (§6.1.2), or extending to cross-modal situations (discussed in next paragraph). We adopt a Siamese network [17] with metric learning [81] to perform the matching attack. A Siamese model is characterized by twin networks which accepts distinct inputs ($\Delta\boldsymbol{w}_i$ and $\Delta\boldsymbol{w}_j$ in our case) and is connected by another network to estimate similarity between the individual embeddings produced by the twin networks. In addition, the weights of the twin networks are shared to ensure extremely similar inputs are not mapped to distant embeddings. Our Siamese network (architecture in Fig. 3b) is constructed as : (a) two FC-128 layers with ReLU activations which individually encodes $\Delta\boldsymbol{w}_i, \Delta\boldsymbol{w}_j$ into a 128-dim embedding; (b) $L_1$ distance layer to represent distance between these embeddings; and (c) FC-1 layer with sigmoid activation to predict the match probability. We minimize the binary-cross entropy loss and perform optimization using RMSProp with learning rate $10^{-3}$.

**Cross-modal Matching Attack.** We extend the matching attack to accommodate the situation where the modality of




**(a)** MLP model for re-identification attack

**(b)** Siamese model for matching attack

**(c)** Model for cross-modal matching attack

**(d)** MTL model to predict user $u$ and attribute $a$

**Figure 3: Architectures of attack models**. Dotted lines indicate shared layers.

attacker's prior knowledge (e.g., text) differs from the private data (e.g., visual data) used by the users during training. In such a scenario, parameter updates can no longer be represented in the same space (as in Eq. 4,6). As a result, the cross-modal matching attack performs:

$$f^{\text{cm-mat}} : (\Delta \boldsymbol{w}_{\text{anon}},\ \boldsymbol{\phi}_u) \rightarrow \text{anon} \stackrel{?}{=} u \tag{7}$$

where $\boldsymbol{\phi}_u \in \mathbb{R}^D$ denotes an embedding of the user's prior data $\mathcal{D}_u^{\text{prior}}$. In §6.1.1, we discuss exactly how we obtain such an embedding. The attack model (architecture in Fig. 3c) to estimate the match probability closely resembles the Siamese network for the matching attack. The only modification is replacing the twin networks with two different networks (each with a single FC-128 layer) to map the inputs into a common 128-dim feature space.

## 5 Experimental Setup: Datasets, Tasks, and Models

In this section, we discuss the experimental setup and datasets (summarized in Table 1) used to train and evaluate the collaboratively learnt ML model in an FL setup.

### 5.1 Datasets

We now present the datasets (Table 1, examples in Fig. 4) used to train and evaluate the collaboratively trained models $f_{\boldsymbol{w}}$. We highlight that the datasets used are well-suited since: (a) they are publicly available; (b) samples are annotated with non-private labels (e.g., tv, flower); (c) examples are complex and realistic; and (d) each training example has a notion of "owner" or "user". Property (d) is particularly important in FL scenarios, as it allows us to partition and distribute data on devices based on user identities. Each of the following paragraphs discusses the (i) dataset $\mathcal{D}$; (ii) corresponding task $\mathcal{X} \rightarrow \mathcal{Y}$; and (iii) training model $f_{\boldsymbol{w}} : \mathcal{X} \rightarrow \mathcal{Y}$ to perform the task.

**(i) PIPA.** PIPA [86] is a dataset consisting of ∼37k personal photos uploaded by actual Flickr users (indicated in the `author` field in Flickr photo metadata). To assure certain minimal amount of per-user data, we only use users with at least 100 images, resulting in 33K images over 53 users. We obtain labels for each image by running a state-of-the-art object detector [45] that detects 80 COCO [51] classes, such as umbrella, backpack, and bicycle. To perform reasonable training and evaluation of the multilabel classification task, we use 19 classes (e.g., chair, cup, tv) that occur in approximately >1% of images with high precision. We train a multi-label image classifier `CNN-PIPA-FL` $f_{\boldsymbol{w}} : \mathbb{R}^{224\times224\times3} \rightarrow \mathbb{R}^{19}$, for this dataset in an FL setup. We use the MobileNet [44] architecture designed specifically to be run on mobile devices, as it is a lightweight architecture that strikes a good balance between latency, accuracy and size.

**(ii) OpenImages.** OpenImages [49] is a large-scale public dataset from Google, consisting of 9M Flickr image URLs and weakly labeled image-level annotations across 19.8k classes. To make training feasible, we prune out users with less than 500 images, resulting in 317k images from 327 users annotated with 18 classes (e.g., food, building). Furthermore, images of the same user can cover a wide time span (typically >5 years). Similar to PIPA, we formulate the training of a multi-label image classifier `CNN-OI-FL` based on the MobileNet architecture.

**(iii) Blog Authorship.** The Blog Authorship Corpus [69] contains ∼681K posts collected from 19K bloggers from

| Dataset ($\mathcal{D}$) | Type | Task | # Users | # Examples | Input ($\mathcal{X}$) | Output ($\mathcal{Y}$) | Model ($f_{\boldsymbol{w}}$) |
|---|---|---|---|---|---|---|---|
| PIPA [86] | Visual | Multi-label classification | 53 | 33,051 | Image | Labels | `CNN-PIPA-FL` |
| OpenImages [49] | Visual | Multi-label classification | 327 | 317,008 | Image | Labels | `CNN-OI-FL` |
| Blog [69] | Language | Language Modeling | 55 | 454,090 | Text | Text | `NNLM-FL` |
| Yelp [20] | Language | Sentiment Analysis | 118 | 85,615 | Text | Score | `NNSA-FL` |

**Table 1: Datasets $\mathcal{D}$ and Models $f_{\boldsymbol{w}}$.** List of datasets used along with corresponding statistics, tasks, and models

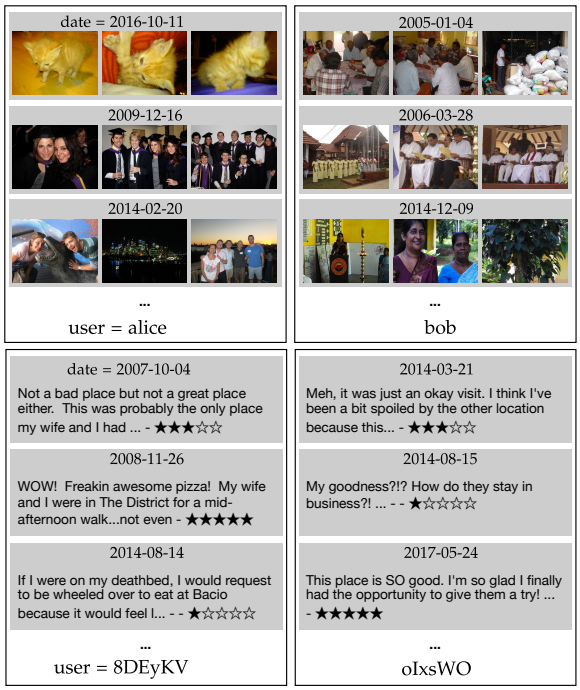



**Figure 4: Examples of users and corresponding data.** OpenImages (top) and Yelp (bottom). Images here are grouped by the anonymized userid and captured/review date. Qualitatively, we observe that the difference between users' data is typically subtle.

`blogger.com`. We work with a subset of 55 users with at least 1000 corresponding posts. Since these blog posts are lengthy (13.5 sentences, 209 words per post), we further split each post into corresponding sentences. As a result, we obtain 454K text sequences over 55 users. We train a language model (`NNLM-FL`): $P(\boldsymbol{x}_t|\boldsymbol{x}_{t-i},\cdots,\boldsymbol{x}_{t-1};\ \boldsymbol{w})$ i.e., predicting probability distribution of the next word $\boldsymbol{x}_t$ in a sequence given contextual information. Language models trained in an FL architecture are currently deployed to enable smart compose keyboards [83]. We train a Neural Network Language Model [9] using an embedding layer (with $E$=100 dims), LSTM layer [42] (with $L$=64 hidden units), and a fully-connected layer (with vocabulary size $V$=5000).

**(iv) Yelp.** The Yelp Dataset [20] contains ∼6M user-reviews of 188K businesses. To allow for each user contributing meaningful parameter deltas, we filter users with at least 500 total reviews. This results in 85K user reviews over 118 users. Each user review contains text (mean length = 180 words) and a 1-5 star rating. We train a sentiment analyzer, modeled as a neural network regressor: $y = f_{\boldsymbol{w}}([\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots])$, where $y \in [1,5]$ is the rating and $\boldsymbol{x}_i$ is a representation of $i$-th word in the review. We use a standard recurrent neural network architecture with an embedding size of $E$=50, $L$=128 hidden LSTM units, and a vocabulary size of $V$=1000.

## 5.2 Data Setup for Adversarial Knowledge

The datasets collected (Table 1) contain sets of user-specific data $\mathcal{D}_u = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_u}$ over users $u \in \mathbb{U}$. A limited subset of this data is strategically held-out to model the adversary's prior knowledge $\mathcal{D}_u^{\text{prior}}$, and the remaining used as the users' private training data $\mathcal{D}_u^{\text{private}}$. We consider multiple prior-data limitation strategies to systematically study their influence on deanonymization attacks: (i) limiting the subset of users the adversary has prior knowledge on (§5.2.1); and (ii) limiting the amount and quality of prior knowledge (§5.2.2).

### 5.2.1 User Scenarios

To tackle the case where a subset of participating users in FL may or may not be a part of adversary's prior knowledge database, we set-up two scenarios:

**Closed-world.** The adversary has some prior information on all users participating anonymously in FL. Consequently, deanonymization of a particular device always maps to a closed-set of 'seen' users. This scenario captures instances of silo-based federated learning scenarios, which typically involve a small number of organizations (the users).

**Open-world.** We extend the above world to additionally include 'unseen' users during FL, for which the adversary does not have prior information. Hence, a parameter delta $\Delta\boldsymbol{w}_{\text{anon}}$ could map either to a seen or an unseen user. This presents a challenging scenario, as it leads to 'finding a needle in a haystack' i.e, the adversary wants to re-identify a particular target user in spite of background noise generated by many unseen users.

| | PIPA | |
|---|---|---|
| split | `random` | `chrono` |
| `CNN-PIPA-FL` | 45.1 | 37.7 |
| `CNN-PIPA-SGD` | 49.7 | 40.7 |
| K-NN | 14.9 | 15.8 |
| Chance | 9.5 | 9.7 |

| | OpenImages | |
|---|---|---|
| split | `random` | `chrono` |
| `CNN-OI-FL` | 62.9 | 62.2 |
| `CNN-OI-SGD` | 68.0 | 67.8 |
| K-NN | 9.7 | 13.6 |
| Chance | 6.3 | 6.3 |

| | Blog | |
|---|---|---|
| split | `random` | `chrono` |
| `NNLM-FL` | 28.02 | 27.83 |
| `NNLM-SGD` | 28.62 | 28.22 |
| Chance | 0.09 | 0.09 |

| | Yelp | |
|---|---|---|
| split | `random` | `chrono` |
| `NNSA-FL` | 0.716 | 0.708 |
| `NNSA-SGD` | 0.576 | 0.602 |
| Chance | 1.472 | 1.514 |

**Table 2: Evaluation of $f_{\boldsymbol{w}}$.** Datasets from Table 1. Metrics used are: (a) PIPA: Average Precision (AP) (b) OpenImages: Average Precision (AP) (c) Blog: Top5 accuracy (d) Yelp: Mean Absolute Error (MAE). For (a-c), higher is better and for (d), lower is better.

#### 5.2.2 Type of Prior Knowledge

To understand the role of prior information in a systematic manner, we consider both the *amount* and *distribution* of adversary's prior information w.r.t private data on the FL device. Specifically for the distribution, we model both $\mathcal{D}_u^{\text{prior}}$ and $\mathcal{D}_u^{\text{private}}$ to be sampled (without replacement) from user $u$'s universal data distribution $\mathcal{D}_u$ in one of the four following manners.

**(i) `random` prior:** Both the prior and private data are IID samples from $\mathcal{D}_u$ i.e., $\mathcal{D}_u^{\text{prior}}, \mathcal{D}_u^{\text{private}} \overset{\text{iid}}{\sim} \mathcal{D}_u$ This scenario captures the adversary scraping information on target user $u$ randomly from various social media sources.

**(ii) `chrono` prior:** We also consider both prior and private data to be sampled non-IID from $\mathcal{D}_u$ by factoring in timestamps of data (e.g., from image EXIF metadata). Here, data in $\mathcal{D}_u^{\text{prior}}$ chronologically precedes data in $\mathcal{D}_u^{\text{private}}$ For instance, this could occur when an adversary has historical data on the targeted user, such as from a previously de-identified account. In the specific case of the PIPA dataset, where the exact timestamp per example is unavailable, we sample prior and private data non-IID using album information (`photoset` field).

**(iii) `profile` prior:** We briefly address a scenario where the adversary uses a set of curated 'profile' data as a proxy to users' data. For instance, by curating targeted prior data $\mathcal{D}_u^{\text{prior}}$ to specifically contain weapons to identify participating users who fit that profile.

**(iv) `cross-model` prior:** We consider the case where adversary's prior data of the user $\mathcal{D}_u^{\text{prior}}$ is gathered from a different modality compared to the private data. For instance, where the prior data is text-based, but the users train on visual data.

### 5.3 Collaborative Models: Training and Performance

In Section 5.1, we discussed details on the datasets and corresponding model architectures $f_{\boldsymbol{w}}$. Section 5.2 presented how we strategically hold-out a subset of the data to serve as adversary's prior knowledge. Now we discuss setup and performances of collaborative models in our FL setting.

**Training Models $f_{\boldsymbol{w}}$.** For each dataset, we train models $f_{\boldsymbol{w}}$ using `FederatedAveraging` (Algorithm 1) [54]. For all models, crucial hyper-parameters (e.g., size of vocabulary or embedding) were selected carefully after rigorous evaluation over a set of standard choices. In `FederatedAveraging` algorithm, we use $C$=0.1 and $E$=1, which we empirically find results in a good trade-off between convergence and communications required. We train the models for 200 epochs with learning rate $\eta$=0.01, resulting in 1-4 GPU days to train a single model for a particular architecture, dataset and scenario. All models are written in Python using the Keras [22] library with a TensorFlow [1] back-end.

Each user $u$ in our datasets is associated with a variable number of examples $\mathcal{D}_u$ sampled according to some distribution (e.g., `chrono`; see §5.2.1). By default, we place half of the users' data $\mathcal{D}_u$ on their anonymous device and reserve the remaining to be used as adversary's prior knowledge. In Section 6.2.1, we vary the size of the adversary's prior knowledge and find attacks possible even in severely data-limited settings (e.g., 1-50 prior samples).

**Evaluation of $f_{\boldsymbol{w}}$.** We evaluate performance of the collaboratively-trained models on a 20% held-out test set. For reference, we similarly evaluate models trained in a centralized manner i.e., standard training from a single pool of training data. The performances of FL-trained models (represented as '`X-FL`') and SGD-trained models ('`X-SGD`') are presented in Table 2. When possible, we also present the $K$-Nearest Neighbours (KNN, with $K$=10) baseline. We observe strong performances of the FL-trained models $f_{\boldsymbol{w}}$ across all datasets, where they consistency recover $80-98\%$ performance of models trained using centralized SGD.

## 6 Evaluation

In the previous section, we discussed training ML models in an FL setup for four different datasets covering various tasks such as image classification and language modeling. Within this FL scenario, we now detail the training of *deanonymization attack* models (§4.3), evaluate their effectiveness, and work towards understanding how the parameter updates leak user-identifiable information.

**Evaluation Metrics.** We use the following metrics (computed using scikit-learn [64]) to evaluate the adversary's attack performance: (i) **Mean Average Precision (AP)**: Adversary's precision-recall curves for held-out user data is com-

| | PIPA (#Users $U$ = 53) | | | | | | OpenImages ($U$ = 327) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | `random` | | | `chrono` | | | `random` | | | `chrono` | | |
| | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 |
| MLP | 91.0 (48×) | 84.7 | 96.3 | 42.2 (22×) | 40.0 | 68.8 | 53.7 (175×) | 51.9 | 77.9 | 32.5 (106×) | 31.9 | 57.1 |
| SVM | 81.3 (43×) | 89.3 | 91.9 | 27.7 (15×) | 43.7 | 49.6 | 49.0 (159×) | 66.5 | 67.0 | 24.6 (80×) | 41.7 | 42.5 |
| kNN | 85.4 (45×) | 82.6 | 92.6 | 31.5 (17×) | 38.4 | 54.8 | 46.0 (150×) | 49.2 | 63.9 | 25.1 (82×) | 30.3 | 43.1 |
| Chance | 1.9 (1×) | 2.0 | 9.9 | 1.9 (1×) | 2.0 | 9.9 | 0.3 (1×) | 0.3 | 1.5 | 0.3 (1×) | 0.3 | 1.5 |

| | Blog ($U$ = 55) | | | | | | Yelp ($U$ = 118) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | `random` | | | `chrono` | | | `random` | | | `chrono` | | |
| | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 |
| MLP | 52.9 (29×) | 50.1 | 89.9 | 44.8 (25×) | 47.6 | 81.3 | 23.5 (28×) | 25.2 | 50.1 | 16.0 (19×) | 18.9 | 38.9 |
| SVM | 35.7 (20×) | 46.3 | 49.2 | 27.0 (15×) | 42.1 | 46.0 | 25.9 (31×) | 43.2 | 44.9 | 17.1 (20×) | 33.3 | 36.7 |
| kNN | 35.6 (20×) | 39.8 | 64.9 | 29.5 (16×) | 35.6 | 58.3 | 21.6 (25×) | 25.3 | 41.1 | 15.4 (18×) | 21.0 | 32.9 |
| Chance | 1.8 (1×) | 1.7 | 8.8 | 1.8 (1×) | 1.6 | 8.8 | 0.9 (1×) | 0.8 | 4.1 | 0.9 (1×) | 0.9 | 4.3 |

**Table 3: Re-identification Attack Evaluation ($\Delta \boldsymbol{w}_{\text{anon}} \rightarrow u$).** Performed in a closed-world. Chance-level AP $\approx 1/U$.

puted. We then compute the per-user Average Precision (area under the precision-recall curves). We report the mean of Average Precisions across users in percentages (i.e., AP×100); (ii) **Increase over Chance**: In order to analyze adversary's information gain, we compute this as (predicted AP)/(chance AP). We display this alongside AP scores in the form: □×; and (iii) **Top-1 accuracy**: We compute the classification success rates over all parameter updates in the test set. These metrics are common among classification tasks e.g., [25, 51, 80] for AP and [24, 36, 50] for Top-1 accuracy. We use the AP as the primary metric, since it also takes into account ranking among predicted classes.

**Training and Evaluation Data for Attacker $f^{\text{adv}}$.** We train the ML models ($f_{\boldsymbol{w}}$ in Table 1) in an FL system simultaneously using two disjoint sets of devices per user: (a) $\mathbb{K}_{\text{anon}}$: anonymous user devices (that adversary wants to deanonymize); and (b) $\mathbb{K}_{\text{prior}}$: adversary's shadow devices containing target users' prior information (that we use to generate training data for attack models in §4.3). For simplicity, we restrict each of these sets to contain a single user. During training of $f_{\boldsymbol{w}}$ over multiple rounds, we accumulate the parameter updates $\Delta \boldsymbol{w}_k^t$ communicated by all devices in FL. To train the attack models $f^{\text{adv}}$, we use the set of parameter updates $\{(\Delta \boldsymbol{w}_k^t, u) : k \in \mathbb{K}_{\text{prior}}\}$, where we know a priori the device $k$ to user $u$ mapping. We discuss in detail training data-limited adversaries in Section 6.2.1. We evaluate attacks on the disjoint set of parameter updates $\{\Delta(\boldsymbol{w}_k^t, u) : k \in \mathbb{K}_{\text{anon}}\}$.

**Representing $\Delta \boldsymbol{w}_k^t$ for Attacks.** The parameter updates contain hundred thousands to millions of parameters. To enable faster training and evaluation of attack models, we choose a subset of parameters by representing $\Delta \boldsymbol{w}_k^t$ using weights of layers which achieves best attack performance: (i) `CNN-PIPA-FL, CNN-OI-FL`: Fully Connected Layer (19K parameters); (ii) `NNLM-FL`: LSTM layer (10K parameters); and (iii) `NNSA-FL`: Embedding layer (50K parameters). This has little impact to our attack; influence of each layer is discussed in Section 6.2.2. Furthermore, we flatten $\Delta \boldsymbol{w}_k^t$ into a vector and $L_2$ normalize it.

## 6.1 Effectiveness of Deanonymization Attacks

In this section, we validate effectiveness of the deanonymization attacks. We begin by understanding the effectiveness in relation to adversary's prior knowledge (§6.1.1 and §6.1.2) and discuss how it can be coupled with attribute inference attacks (§6.1.3).

### 6.1.1 Impact of Adversary's Prior Distributions

In this section, we focus on how *types* of adversary's prior knowledge (§5.2.2) influences effectiveness of deanonymization. Consequently, we address a range of scenarios, such as when the adversary has similar (`random`) or historical prior data (`chrono`) of the targeted users to perform deanonymization. We also evaluate the novel challenge where the prior data is from a different modality (`cross-modal`).

**Leveraging `random` and `chrono` prior to deanonymize.** We present key results of the re-identification attack model 'MLP' (§4.3): $f^{\text{re-id}} : \Delta \boldsymbol{w}_{\text{anon}}^t \rightarrow u$ (in a closed-world setting) in Table 3. In addition, as baseline attack methods, we also demonstrate performances of 'SVM' (a linear support vector machine) and 'kNN' (a $k$-nearest neighbour classifier using $k$=10).

From the results presented in Table 3, we observe: (i) All deanonymization attacks greatly outperform chance-level performances, with as much as 175× boost for MLP on the OpenImages dataset under the `random` prior, highlighting the effectiveness of the proposed deanonymization attack; (ii) Even the most simple K-NN attack is reasonably effective and already presents a significant threat (150× over random chance

on OpenImages, `random` prior); (iii) MLP is highly effective across all datasets and splits (175× over random chance on OpenImages, `random` prior); (iv) Although the absolute AP scores are lower for the more challenging and larger OpenImages dataset (53.7% AP on `random` prior), the increase over chance level performance is significantly higher (48× on PIPA vs. 175× on OpenImages under the same `random` prior); (v) The attack is effective (19-106×) even on `chrono` priors, where the adversary uses historical prior information to deanonymize users.

The above experiments were performed in a non-IID data-distribution among devices, which is natural in FL since users participate with personal data exhibiting unique biases (§4.2). We also perform attack evaluation in a contrasting IID setup, where we manually unbias data on devices by replacing each user example with an example drawn IID from $\mathcal{D} = \bigcup_k \mathcal{D}_k$. We observed near-chance-level adversary performance (e.g., 1.5× chance-level for PIPA) since user data is no longer characteristic. There is strong evidence that anonymous model parameter updates contain ample user information in an FL setup that allows for effective deanonymization.

**Cross-modal attacks**. We now evaluate the effectiveness of deanonymization attacks with a `cross-modal` prior (Section 5.2.2). Here, the adversary is limited to prior knowledge from a *different* modality from the data used during training by the users. In particular, we consider the case where the prior data consists of text samples and the private data consists of images. As we are not aware of any dataset which provides cross-modal user-generated data to evaluate the attack, we substitute PIPA prior image samples with corresponding text-representations obtained using a Neural Image Caption generator [78]. Using this setup, we train the cross-modal matching network $f^{\text{cm-mat}} : (\Delta \boldsymbol{w}_{\text{anon}},\ \boldsymbol{\phi}_u) \rightarrow \text{anon} \stackrel{?}{=} u$ (Eq. 7). To obtain a compact text representation $\boldsymbol{\phi}_u$ over the prior knowledge (set of text sentences for a particular user), we: (i) obtain the 4096-dim sentence-level embedding using InferSent [23]; and (ii) compute the mean over the sentence embeddings for the user. We evaluate $f^{\text{cm-mat}}$ on a balanced set of 10K pairs $\{((\Delta \boldsymbol{w}_{\text{anon}},\ \boldsymbol{\phi}_u), \mathbb{1}_{\text{anon}=u})\}$. We observe an attack performance of 76.3 AP (chance = 50.0 AP), indicating that model updates can be interestingly deanonymized even using data from another modality.

**Attacking using `profile` prior**. In the previous attacks we looked at the task of deanonymizing devices by associating the parameter updates to prior data of users. We now look at a slightly different task of linking devices that fit a certain `profile` prior. We achieve this by manually constructing $\mathcal{D}^{\text{profile}}$ to comprise of examples of interest e.g., weapons. In Figure 5 we display the top users (in the OpenImages dataset) found using the re-identification attack who fit the corresponding profiles. We observe: (i) devices can be remarkably singled out using various proxy distributions (of e.g., handgun, guitar) circumventing the need for real user data; (ii) however, valid correlations in data can sometimes lead to false positives. For instance, 'dumbbells' which often co-occur in images along with other physical equipment devices leads to bicycle images of user 128 (which also displays similar correlations) being falsely identified.


**Figure 5: `profile` prior.** Devices can be isolated using proxy distributions of certain profiles e.g., guitars. Rows denote private data $\mathcal{D}_u^{\text{private}}$ of users on devices.

### 6.1.2 Impact of Number of Seen and Unseen Users

In the previous section, we evaluated attacks in a closed-world scenario (§5.2.1), where the adversary was aware of every users' existence (i.e., included in prior knowledge). We now consider the open-world scenario, where at test-time the adversary additionally encounters model updates generated by *unseen* users (i.e., not in the prior knowledge). This introduces the challenge of differentiating between seen and unseen identities when deanonymizing.

**User Split**. In our experimental setup, we split the users $\mathbb{U}$ into three variably-sized disjoint sets: (a) $\mathbb{U}_{\text{unseen}}$: prior data is unavailable and should be classified as unseen at test-time; (b) $\mathbb{U}_{\text{seen}}$: prior data is available and should be deanonymized at test-time; and (c) $\mathbb{U}_{\text{holdout}}$: these users are reserved purely for training purposes.

**Re-identification Setup**. Previously in the closed-world scenario, we trained the MLP (§4.3) classifier $f^{\text{re-id}} : \Delta \boldsymbol{w}_k \rightarrow u$ with $|\mathbb{U}|$ classes representing all users at test time. Now we train a similar classifier over $|\mathbb{U}_{\text{seen}}|+1$ output classes with the additional class `unseen` collectively denoting unseen users. During training, we use users $\mathbb{U}_{\text{holdout}}$ and their parameter updates to train the `unseen` class.

**Matching Setup**. We train a Siamese network (§4.3) using parameter updates from held-out and seen set of users. Given a pair $(\Delta \boldsymbol{w}_i, \Delta \boldsymbol{w}_j)$, the network predicts the probability $\mathbb{P}[i = j]$ of being generated by the same user.

**Evaluation**. The performances are evaluated at different

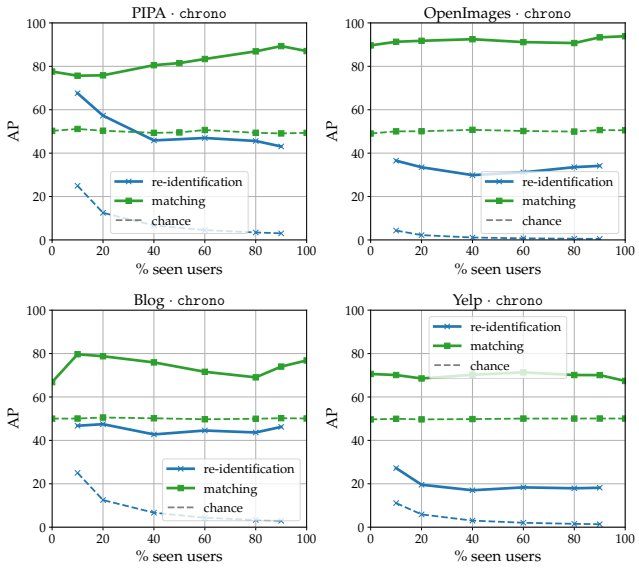



**Figure 6: Open-world evaluation.** Across re-identification (MLP) and matching (Siamese) attack models.

ratios of seen and unseen users at test time. We keep the size of the hold-out set constant to one-third of the total number of users. Evaluation for both re-identification and matching tasks on the challenging `chrono` prior distributions per dataset are presented in Figure 6. We observe: (i) even in the open-world scenario, we perform much higher than chance-level for both the tasks consistently across a wide range of seen vs. unseen scenarios; (ii) for the re-identification attack, as % seen users increase, the complexity of the task increases as well (due to larger output-space). Hence, we notice a drop in AP performance (67%→43% in PIPA). However, performance compared to chance-level significantly increases (3× →14×); (iii) in the matching task, the Siamese model performs much higher than chance-level even in a purely open-world setting, with no seen users (1.5× for PIPA and 1.8× for OpenImages). We find both the re-identification and matching attacks generalize well in the presence of unseen users at test time.

#### 6.1.3 Amplification with Attribute Inference Attacks

We now discuss how deanonymization attacks can be coupled with related inference attacks on model updates. Specifically, we consider the recent attribute inference attack [56], which recovers sensitive properties (e.g., race) that holds for subsets of training data. In this particular case, our attack objective involves jointly inferring both identity (via our deanonymization attacks) and sensitive attributes (via attribute inference attacks) via transmitted model updates.

To evaluate the attacks, we closely follow the data setup on Melis et al. [56] on the PIPA dataset. Attribute inference in this setting involves inferring sensitive attributes (e.g., age) from the model updates. To this end, we first train individual attribute classification models for each of the five attributes, and an additional re-identification model. All the classification models are MLPs following the architecture of the re-identification model. Table 4 (column STL) presents results over the five attribute inference (column AttrInf) tasks and deanonymization (column Deanon). Here, we observe that an attacker can consistently achieve 85.2-98.5% accuracy in inferring various attributes from model updates and 87.6% accuracy in inferring identities of participants. These results suggest that model updates indeed leak details unrelated to the trained task (recognizing chair, couch, etc.) and allows an attacker to recover sensitive attributes of the users' training data (via attribute inference) and further link them to an identity (via deanonymization).

| | | STL | | MTL | |
|---|---|---|---|---|---|
| Attributes | # Attrs | AttrInf | Deanon | AttrInf | Deanon |
| Age | 5 | 89.1 | - | 90.8 | 90.9 |
| Gender | 2 | 93.1 | - | 94.4 | 91.6 |
| Glasses | 3 | 98.5 | - | 98.9 | 91.3 |
| Hair Color | 3 | 85.2 | - | 88.7 | 90.1 |
| Hair Length | 5 | 91.3 | - | 91.3 | 90.1 |
| - | - | - | 87.6 | - | - |

**Table 4: Attribute Inference and Deanonymization Attack Performances.** Results are reported in top-1 accuracies. Columns indicate when the inference tasks are trained individually (STL) and jointly (MTL).

We now recast the problem of inference on attributes and identities as a multi-task learning (MTL) [19] problem. The core idea is to exploit commonalities between the two related tasks to learn a better representation jointly benefiting the tasks. To achieve this, we extend our re-identification model (§4.3, which performs user classification) with a secondary classification head (which performs attribute inference; see Fig. 3d). Consequently, the model is simultaneously trained for both attribute inference and deanonymization using their corresponding losses. The results for the model is presented under the MTL column in Table 4. We observe by learning the two tasks jointly can improve attribute inference performances consistently by 0-3.5% and deanonymization by 2.5-4%. Our results suggest that apart from jointly inferring sensitive attributes and recovering identities, the two related attacks surprisingly amplify each other's performances.

### 6.2 Analysis

In this section, we take a closer look at various factors that influence (e.g., amount of training data) the effectiveness of attacks. For simplicity, we study the factors using the re-identification attack in a closed-world setup. We conclude the section by reasoning why model updates lend themselves to deanonymization risks.

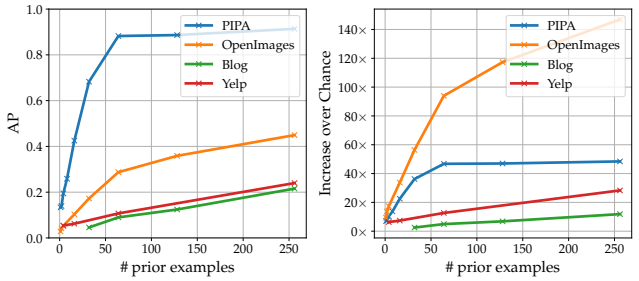



**Figure 7: Number of prior examples per user.** Evaluated on closed-world re-identification.

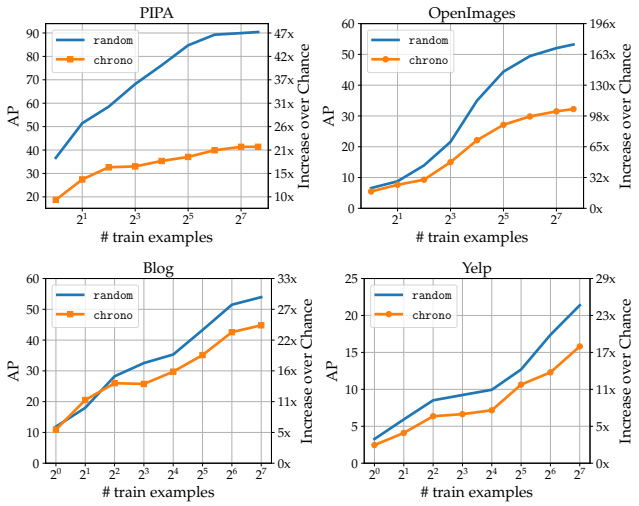



**Figure 8: Number of training examples per user.** Evaluated on closed-world re-identification.

### 6.2.1 Amount of Training Data

We study the influence of data-limitation in deanonymization attacks in a closed-world re-identification scenario. We previously used the entire reserve set of prior information to perform the deanonymization attacks. We first address the influence in the amount of this prior information available per target user. From Figure 7, we observe: (i) even a single prior example of the user leads to non-chance-level re-identification, with as much as 13.4% AP (7 ×) performance on PIPA; (ii) performance of the attack increases significantly with the size of prior knowledge across all datasets e.g., 67% increase in performance on OpenImages by using 16→32 prior examples; (iii) some tasks require more prior information than others. For instance, although Blog and PIPA contain similar number of users, an adversary requires approximately 5× as many prior Blog examples to achieve 20% AP. We attribute this to a weaker signal generated from sparse text content in Blog, as compared to dense pixel content in PIPA.

We also address the impact of size of training set ($\{\Delta \boldsymbol{w}_k^t : k \in \mathbb{K}_{\text{anon}}\}$) for attack models. We train multiple re-identification MLP adversary models, each trained on a random subset of training data with increasing sizes. In Figure 8, we observe an adversary can train reasonably effective attack models, even with extremely limited labeled data. In particular, attack performances of 3-22× can be obtained with a single labeled example per user. While the amount of data (either training or prior) does strongly influence the attack performance, we nonetheless find deanonymization is possible in strongly data-limited situations.

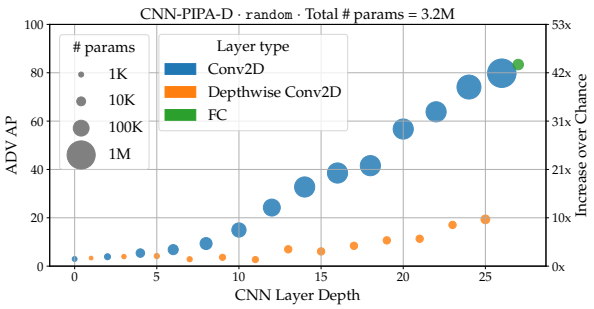



**Figure 9: Re-identification performance by depth.** Bubble sizes indicate the number of parameters in each layer. Last two layers contains 1M and 19K parameters respectively.

| | | NNLM-D (92K) | | NNSM-D (141K) | |
|---|---|---|---|---|---|
| Depth | Layer type | AP | # params | AP | # params |
| 1 | Embedding | 15.7 (9×) | 50K | 23.5 (28×) | 50K |
| 2 | LSTM | 46.0 (25×) | 10K | 19.2 (23×) | 91K |
| 3 | FC | 38.8 (21×) | 32K | 17.6 (21×) | 128 |

**Table 5: Re-identification performance by depth**. For models trained on Blog and Yelp.

### 6.2.2 Impact of Parameter Layers

The deanonymization targets (i.e., model updates $\Delta \boldsymbol{w}$) comprise of parameters from multiple layers of a deep neural network. We now analyze how the layer type and depth affect attacker performance, since they influence the type of task-specific information learnt by the model. For instance, in CNNs, layers at various depths of the network are known to learn various concepts [85] – lower level features (e.g., corners, edges) in the initial layers and higher level features (e.g., wheel, bird's feet) in the final layers. For parameters updates contributed by each individual layer, we train a total of 27 attack models for CNN-based models and 3 attack models for LSTM-based models. We were limited by storage capacity to evaluate on OpenImages as it would require > 3TB.

From layer-wise performances in Figure 9 and Table 5, we observe: (i) *all* layers provide above-chance level information to perform re-identification attacks; (ii) in the CNN model, higher level layers contain more identifiable information with the final fully connected (FC) layer being the most informa-

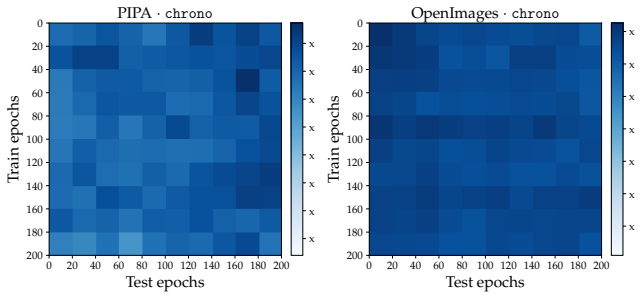



**Figure 10: Effect of the epoch $t$.** On the re-identification attack $\Delta \boldsymbol{w}^{t}_{\text{anon}} \rightarrow u$. As an example, the top-right cell denotes when the MLP was trained on $\Delta \boldsymbol{w}^{t}_{u}, t \in [0, 20]$ and evaluated on $\Delta \boldsymbol{w}^{t'}_{\text{anon}}, t' \in [180, 200]$

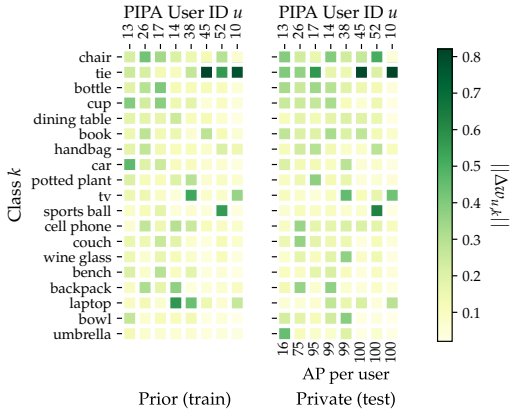



**Figure 11: User bias visualized on parameter updates.**

tive; (iii) in the RNN-based models, the LSTM parameters are more informative for language modeling, whereas it is the embedding layer for sentiment analysis.

### 6.2.3 Impact of Optimization State

We now analyze the influence of training progress of the ML model on deanonymization attacks. We group the parameter updates (separately for train and test attack sets), based on the epoch ranges during which they were generated. We split parameter updates collected during training of $f_{\boldsymbol{w}}$ over 200 epochs into 10 ranges, each with 20 epochs. We train and evaluate the MLP re-identification attack model over all $10 \times 10$ train-eval pairs. From Figure 10, we observe that the training progress at which the update was generated has little influence on the performance indicating an adversary can re-identify users at any stage of training.

### 6.2.4 Reasoning About Effectiveness of Attacks

In Section 4.2 (Fig. 2), we observed that users display a bias resulting in lower variations in data they capture. Consequently, we conjectured that the resulting bias is consistently encoded in the parameter updates, even when they are computed on different (prior and private) sets of users' data. To validate, we take a closer look at the parameter updates $\Delta \boldsymbol{w}^{\text{prior}}_{u}, \Delta \boldsymbol{w}^{\text{private}}_{u} \in \mathbb{R}^{D \times K}$ in the FC layer of eight users in the PIPA FL setup, where $K$ (=19) is the number of classes and $D$(=1024) represents weights per class. In Figure 11, we illustrate bias per user (columns) in the parameter delta space by computing the $L_2$-norm of each of the $K$ class weight vectors (column-dimension in $\Delta \boldsymbol{w}_u$). We observe: (i) for users who can be re-identified highly accurately (e.g., $u$=10), we find that the user is more biased towards images containing 'tie', 'tv', and 'laptop'. Furthermore, this bias is consistent in both the user's prior and private update signals; and (ii) surprisingly, even when biases are not entirely consistent (e.g., $u$=17), we find attacks to be reasonable effective (AP=95); and (iii) for users who cannot be re-identified easily (e.g., $u$=13), the biases are inconsistent between the prior (biased towards cars and cups) and private (biased towards chairs, ties, and umbrellas) update signals. We find our conjecture that the user bias signal translates to the parameter delta space, holds reasonably well, leading to highly effective deanonymization attacks we saw in the previous sections.

## 7 Countermeasures

In the previous section, we evaluated our threat models across a variety of challenging scenarios and consistently observed deanonymization risks. In this section, we present mitigation strategies to counter these attacks.

We attributed (§6.2.4) the effectiveness of the attacks to user bias, which is a powerful statistical signal in both the limited set of adversary's prior data and the users' private data. The focus of our mitigation strategies is to perturb the data bias on the anonymous device, to provide a false signal to the adversary. We spell out our requirements for the defense as: (a) maximally retain utility (performance of $f_{\boldsymbol{w}}$); (b) involve low computation overhead; (c) not rely on a trusted-third party; and (d) allow users to selectively employ the strategy to various extents depending on personal preferences.

## 7.1 Methods

Based on the requirements, we propose data-centric mitigation strategies: devices adversarially bias their data distribution on devices, rather than directly perturb model parameters. More specifically, users mix their original data $\mathcal{D}_u$ with certain "background" data $\mathcal{D}^{\text{bkg}}$ to "blend into the crowd", thereby rendering the parameters less user-specific. Here, the mixing takes place prior to participation in FL.

**Collecting $\mathcal{D}^{\text{bkg}}$.** The background dataset $\mathcal{D}^{\text{bkg}}$ can be any large (labeled) set of training examples for the same federated learning task (e.g. user-annotated dataset, scraped data from

| $\mathcal{D}$ | Source ($\mathcal{D}$) | $\mathcal{D}^{\text{bkg}}$ | Source ($\mathcal{D}^{\text{bkg}}$) | $\lvert\mathcal{D}^{\text{bkg}}\rvert$ |
|---|---|---|---|---|
| PIPA [86] | Flickr | OpenImages | Flickr | 59K |
| OpenImages [49] | Flickr | OpenImages | Flickr | 490K |
| Blog [69] | Blogger | WikiReading [40] | Wikipedia | 3M |
| Yelp [20] | Yelp | Amazon Reviews [37] | Amazon | 1.7M |

**Table 6: Background datasets and sources.** Used to mitigate deanonymization attacks.

the Internet, a trusted open-source dataset). The background datasets used in our experiments, their sources and sizes are listed in Table 6. We only select a random subset of the original background datasets (s.t. $|\mathcal{D}^{\text{bkg}}| \gg |\mathcal{D}_u|$) in each case, for experiments to complete within a feasible amount of time. The preprocessing of $\mathcal{D}^{\text{bkg}}$ and $\mathcal{D}$ are identical.

Now, we present three countermeasures which alter the characteristic data-distribution of the users.

**Data Replacement (`bkg-repl`).** Each user replaces a fraction $\alpha \in [0,1]$ of his/her data $\mathcal{D}_u$ with ones from $\mathcal{D}^{\text{bkg}}$. At $\alpha = 0$, no mitigation strategy takes place; at $\alpha = 1$, every user has identical data composition. However, the strategy skews FL to learn from a noisy background data distribution displaying different statistics, instead of learning from interesting user data on which evaluation metrics need to be maximized.

**Data Augmentation (`rand-aug`).** Instead of replacing, the user *augments* random data (since more data helps [34, 73]) from $\mathcal{D}^{\text{bkg}}$:

$$\hat{\mathcal{D}}_u \leftarrow \mathcal{D}_u \cup \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim \mathcal{D}^{\text{bkg}}\}_{i=1}^{\alpha \cdot |\mathcal{D}_u|}, \tag{8}$$

where $\alpha \geq 0$ determines the size of augmentation. As $\alpha \to \infty$, devices' empirical data distributions converge to $\mathcal{D}^{\text{bkg}}$, making them indistinguishable from each other.

**Mode-specific Data Augmentation (`mm-aug`).** So far, the users' strategies were to mix their data with background data from a single source $\mathcal{D}^{\text{bkg}}$. We now consider the strategy where each device mixes data from *different* topics i.e., modes of the data distribution. For instance, Alice adversarially adds sports content to her data to mask her interest in automobiles before participating in FL. We perform this by first clustering $\mathcal{D}^{\text{bkg}}$ into $M$ clusters $\bigcup_{m=1}^{M} \mathcal{D}_m^{\text{bkg}}$. We use the k-means clustering over the ImageNet pretrained Mobilenet features. Each user $u$ picks a cluster $m$ at random, and augments its data with ones from the cluster:

$$\hat{\mathcal{D}}_u \leftarrow \mathcal{D}_u \cup \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim \mathcal{D}_m^{\text{bkg}}\}_{i=1}^{\alpha \cdot |\mathcal{D}_u|} \tag{9}$$

where $\alpha \geq 0$ controls the degree of mix. We use $M$=100 for PIPA, $M$=500 for OpenImages, $M$=300 for Blog and Yelp.

We additionally consider two perturbation-based baselines to our data-augmentation strategies.

**DP-FederatedAveraging (`dp-fedavg`).** We implement a differentially private variant [55] of the Federated Averaging algorithm. They key idea is to provide $(\varepsilon, \delta)$ participant-level differential privacy guarantees by bounding the contribution (the parameter update) provided by each participant. In practise, the contributions are bounded by clipping the parameter updates and further adding random noise. In our experiments, we fix the clipping value to 50 and vary magnitude of gaussian noise added during training.

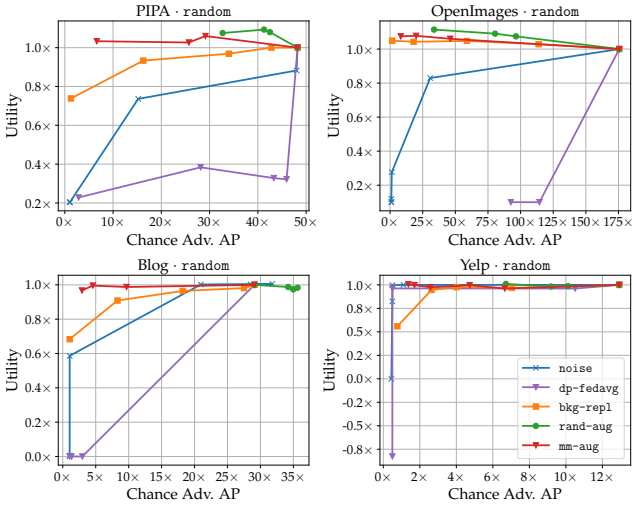



**Figure 12: Mitigation strategies evaluation.** Re-identification AP obtained by varying $\alpha$ and $\sigma^2$ in closed-world scenario. Top-left is the ideal region. Higher $\alpha$ and $\sigma^2$ values pushes operating points towards the left (i.e., lower deanonymization performance).

**Random Perturbations (`noise`).** Although `dp-fedavg` has shown success in large-scale scenarios (with thousands of users), we found difficulty achieving reasonable results in our setup. Hence, we consider a relaxed version of introducing perturbations, where the user introduces zero-centered Gaussian noise to model updates before leaving the device.

## 7.2 Evaluation

We evaluate the proposed mitigation strategies by measuring the adversary's performance against our countermeasures. We analyze the effectiveness of the defense against the strongest adversary: closed-world re-identification attack on `random` prior (§6.1.1, Table 3).

We evaluate the strategies in terms of trade-off between privacy (reduction in adversary's performance) and utility (decentralized learning performance). As in §6.1.1, we measure the adversary's performance as increase over chance-level AP. We measure utility by performance scores normalized to have utility=1.0 when no mitigation takes place.

The mitigation strategies are evaluated on a curve by varying hyperparameters. For `bkg-repl`, we use $\alpha \in \{0.0, 0.25, 0.50, 0.75, 1.0\}$. For `rand-aug` and `mm-aug`, we use $\alpha \in \{0.0, 0.5, 1.0, 2.0\}$. For `dp-fedavg`, we fix the clip value to 50 and

vary the noise multiplier in the range $[10^{-5}, 10^{-1}]$. For `noise`, we consider Gaussian noise with $\mu = 0$ and $\sigma^2 \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$.

We present evaluation for our strategies in Figure 12. Better mitigation strategies have curves towards the top-left corners in each plot (high privacy, high utility). We observe: (i) the perturbation-based baselines (`noise` and `dp-fedavg`) in most cases severely decreases utility at a small gain in privacy; (ii) replacing data with background samples (`bkg-repl`) is a good alternative strategy: we have both higher privacy and utility than perturbation methods. However, due to a domain-shift between $\mathcal{D}^{\text{bkg}}$ and $\mathcal{D}$, utility is often impacted. This can be observed in PIPA, Blog and Yelp datasets, where it achieves $< 0.75\times$ utility since the user data is no longer used; (iii) the augmentation-based strategies `rand-aug` and `mm-aug` outperforms `noise` and `bkg-repl` in terms of utility and privacy; (iv) for the `mm-aug` strategy, already at $\alpha = 0.5$, we observe a good combination of privacy and utility (75% decrease in adversary's AP in OpenImages, compared to 45% for `rand-aug` and 67% for `bkg-replace`).

We find the strategy `mm-aug` offer the most effective and practical operating points, requiring the user to perform minimal augmentation to achieve reasonable privacy. We remark that the utility for `mm-aug` can be more than 1.0 even at higher privacy level, as can be seen in PIPA and OpenImages. This is due to the effect of additional data [34, 73]. This increased privacy and utility comes at the cost of preparing a labeled dataset and increased training time (training set becomes $(1+\alpha)\times$ bulky). However, this overhead will be less costly with increasingly powerful devices and energy-efficient ML models for mobile devices [44, 68].

## 8 Conclusion

In this paper, we were motivated to understand privacy threats in Federated Learning, which is designed towards large-scale learning on user data on personal devices. We questioned whether devices can truly participate anonymously without compromising the identity of individuals. Our results indicate that the devices can be effectively deanonymized using the transmitted model parameter updates and a reasonable amount of prior data. We found this to be possible due to the inherent user bias in captured data acting as a fingerprint that is consistent across different sets of data captured by the user. To mitigate such attacks, we proposed calibrated domain-specific data augmentation, which shows strong results in preventing deanonymization with minimal impact to utility.

## References


[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[2] Mohammed Abuhamad, Tamer AbuHmed, Aziz Mohaisen, and DaeHun Nyang. Large-scale and language-oblivious code authorship identification. In *CCS*, 2018.

[3] Mishari Almishari and Gene Tsudik. Exploring linkability of user reviews. In *ESORICS*, 2012.

[4] Michael Arrington. Aol proudly releases massive amounts of private data. https://en.wikipedia.org/wiki/AOL_search_data_leak, 2006. Accessed August 29, 2019.

[5] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*. JMLR, 2018.

[6] Michael Backes, Pascal Berrang, Anna Hecksteden, Mathias Humbert, Andreas Keller, and Tim Meyer. Privacy in epigenetics: Temporal linkability of microrna expression profiles. In *USENIX Security Symposium*, 2016.

[7] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948, 2020.

[8] Micheal Barbaro and Tom Zeller Jr. A face is exposed for aol searcher no. 4417749. https://www.nytimes.com/2006/08/09/technology/09aol.html, 2006. Accessed August 29, 2019.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JMLR*, 2003.

[10] James Bennett, Stan Lanning, et al. The netflix prize. In *KDD workshop*, 2007.

[11] Richard A Berk. An introduction to sample selection bias in sociological data. *American sociological review*, pages 386–398, 1983.

[12] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643, 2019.

[13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML PKDD*, pages 387–402. Springer, 2013.

[14] Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

[15] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé M Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. To appear.

[16] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.

[17] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *NeurIPS*, 1994.

[18] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE S&P*, 2017.

[19] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[20] Yelp Dataset Challenge. Yelp dataset challenge, 2013.

[21] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.

[22] François Chollet et al. Keras. https://keras.io, 2015.

[23] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[25] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[26] Barbara Fadem. *Behavioral science in medicine*. Lippincott Williams & Wilkins, 2012.

[27] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 2015.

[28] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS*, 2018.

[29] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. In *NIPS PPML Workshop*, 2017.

[30] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

[31] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. Exploiting innocuous activity for correlating users across sites. In *WWW*, 2013.

[32] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, and Gang Wang abd Xinyu Xing. LEMNA: Explaining Deep Learning based Security Applications. In *CCS*, 2018.

[33] Danna Gurari, Qing Li, Chi Lin, Yinan Zhao, Anhong Guo, Abigale Stangl, and Jeffrey P Bigham. Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 939–948, 2019.

[34] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

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

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[37] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, 2016.

[38] Miguel A Hernán, Sonia Hernández-Díaz, and James M Robins. A structural approach to selection bias. *Epidemiology*, 15(5):615–625, 2004.

[39] Nestor Hernandez, Mizanur Rahman, Ruben Recabarren, and Bogdan Carbunar. Fraud de-anonymization for fun and profit. In *CCS*, 2018.

[40] Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. Wikireading: A novel large-scale language understanding task over wikipedia. In *ACL*, 2016.

[41] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *CCS*, 2017.

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[43] White House. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *White House, Washington, DC*, pages 1–62, 2012.

[44] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[45] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017.

[46] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[47] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS PPML Workshop*, 2016.

[48] Nitish Korula and Silvio Lattanzi. An efficient reconciliation algorithm for social networks. *VLDB*, 2014.

[49] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[51] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[52] Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD*, 2005.

[53] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. https://research.googleblog.com/2017/04/federated-learning-collaborative.html, 2017. Accessed January 21, 2018.

[54] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[55] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

[56] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Inference attacks against collaborative learning. In *S&P*, 2019.

[57] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *S&P*, 2008.

[58] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *S&P*, 2009.

[59] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. In *S&P*, 2019.

[60] Council of European Union. Eu general data protection regulation (gdpr): Article 5, principles relating to processing of personal data. https://gdpr-info.eu/art-5-gdpr/, 2016. Accessed: 2020-06-18.

[61] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Towards a visual privacy advisor: Understanding and predicting privacy risks in images. In *ICCV*, 2017.

[62] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, 2019.

[63] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. SoK: Towards the Science of Security and Privacy in Machine Learning. In *Euro S&P*, 2018.

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

[65] Daniele Perito, Claude Castelluccia, Mohamed Ali Kaafar, and Pere Manils. How unique and traceable are usernames? In *PETS*, 2011.

[66] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *NDSS*, 2018.

[67] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NDSS*, 2019.

[68] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018.

[69] Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. Effects of age and gender on blogging. In *AAAI*, 2006.

[70] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *CCS*, 2015.

[71] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *S&P*, 2017.

[72] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *NeurIPS*, 2017.

[73] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[74] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. In *AMIA*, 1997.

[75] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 1997.

[76] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[77] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.

[78] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[79] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. With Great Training Comes Great Vulnerability: Practical Attacks against Transfer Learning. In *USENIX Security*, 2018.

[80] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*, 2016.

[81] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *NeurIPS*, 2006.

[82] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

[83] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[84] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y. Zhao. Automated Crowdturfing Attacks and Defenses in Online Review Systems. In *CCS*, 2017.

[85] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[86] Ning Zhang, Manohar Paluri, Yaniv Taigman, Rob Fergus, and Lubomir Bourdev. Beyond frontal faces: Improving person recognition using multiple cues. In *CVPR*, 2015.

[87] Xiao Zhang and David Evans. Cost-Sensitive Robustness against Adversarial Examples. In *ICLR*, 2019.