Gradient-Leaks: Understanding and Controlling Deanonymization in Federated Learning

Tribhuvanesh Orekondy¹

Bernt Schiele¹

Mario Fritz²

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 minimal training information (model parameter deltas) computed locally on personal data. However, the degree to which user-specific information is encoded in the model deltas is poorly understood. In this paper, we identify model deltas encode subtle variations in which users capture and generate data. The variations provide a powerful 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 closedand 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

Effectiveness of machine learning (ML) models for a variety of tasks strongly benefits from large-scale datasets used during training [30,67]. So far, services have primarily collected training data from individuals to train such models and make them available for a variety of applications e.g., smart compose keyboards. However, due to increasingly strict privacy regulations and concerns on data collection, Federated Learning (FL) approaches [13,50] (also referred to as Collaborative ML [64]) present a promising avenue. While traditional approaches have brought data to computation, FL seeks to bring computation to the data. Consequently, computation is primarily performed on rich user-generated data (e.g., photos, keystrokes) locally on personal devices such as smartphones.

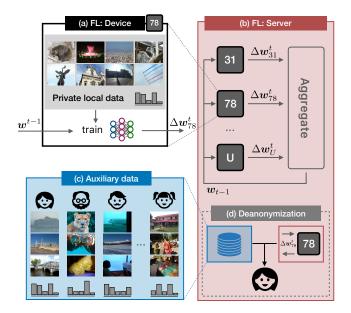


Figure 1: **Deanonymization in Federated Learning.** In this paper, we study how subtle user-biases captured in model parameter deltas leads to deanonymization of their devices.

Only the minimal amount of information (model parameter deltas) necessary to train the ML model is anonymously communicated by the device to an potentially untrusted server. To fully ensure the privacy and anonymity of participating users, it is crucial to understand the privacy leakage in the parameter deltas. In this paper, we seek to identify and understand the leakage that results in deanonymization of individuals.

There exists a long line of work on deanonymization, with research specific to statistical databases dating back to Sweeney [68, 69]. More recently, research has highlighted deanonymization of individuals in anonymized data, such as movie reviews [53], social networks [54], programmatic code [2], product reviews [35], and epigenetics [6]. Insights obtained from studying such risks have played a crucial role in preventing future (and potentially more severe) data breaches. For instance, when Chicago Medical Center offered to share

deidentified patient data with Google, a lawsuit [72] was filed against the parties in response to concerns that Google can potentially link the records to a vast amount of user information it already possesses.

Central to deanonymization attacks is finding a quasiidentification mechanism that allows private data (parameter deltas in our case) to be correlated with auxiliary data that sets the basis for re-identification of individuals. However, since the parameter deltas in our scenario involves extremely highdimensional (100K-10M) stochastic floating-point tensors, we seek to identify a statistical signal that can be utilized instead. We find the subtle variations specific to users when generating training data (e.g., Alice capturing more photos of food) also appears in the transmitted parameter deltas. We seek to demonstrate that this signal can be effectively exploited to deanonymize devices participating in FL.

Due to the anonymous nature of communication in FL [13,31], a bidirectional level of trust is required between the server (who aggregates raw parameter deltas) and the participating users. As a result, server-side deanonymization can be a double-edged sword. On the one hand, the technique allows the server to perform forensics, such as deanonymizing users who exhibit criminal behaviours, or isolating adversarial devices who contribute poisoned data [7,37]. On the other hand, deanonymization also sets precedence for launching more targeted attacks against certain users e.g., model inversion [24,37], membership inference [55,61,65], attribute inference [52], and poisoning [37,55]. While we briefly address the former role in the paper, we predominantly take the role of a malicious server for the remainder of the paper.

Within our setup following the popular Federated Learning architecture [13, 50], participating devices intermittently communicate model parameter deltas (computed on local private data) to a server. Either the protocol [13], or the users' choice to participate with a pseudonym identity, allows the devices (Fig. 1a) to anonymously share these parameter deltas which is aggregated by the server (Fig. 1b). We take the role of a server who intends to deanonymize participating devices (Fig. 1d) with limited access to public information of users (Fig. 1c) which can be obtained by e.g., scraping social network content. We evaluate our attack models (re-identification and matching) in a variety of scenarios (open- and closed-world) and with different distributions and amounts of prior information available on the participants. Furthermore, we perform all evaluation on realworld (anonymized) user-generated data on high-dimensional tasks using state-of-the-art neural networks. In our evaluation, we find simple deanonymization attacks to be highly effective across a range of scenarios. For instance, we find deanonymization of participants is possible with 2-9× chancelevel performance with a single input known a priori and with up to $175 \times$ with the full range of the user's prior information.

To protect against such attacks, we study strategies that ensure local privacy; user-linkable information in parameter deltas is mitigated even before it leaves the client. Our proposed methods complement existing work of enforcing privacy on parameter deltas achieved by introducing randomized perturbations using Differential Privacy Mechanisms [26,51], performing secure multi-party computation [14], and executing homomorphic encryption [27,78]. 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.

2 Related Work

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

Deanonymizing (Insufficiently) Anonymized Data. ganizations 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) [69], Netflix prize dataset [10], and AOL search logs [4]. However, a long of line work, dating back to [68, 69] 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 [69], IMDb movie ratings [53], and search keywords [8] respectively. This has motivated numerous research in the area of identifying factors that potentially lead to deanonymization of individuals, such as in social networks [54], programmatic code [2], and product reviews [35]. Research has also identified various sources of quasi-identifiers in unstructured data: profile attributes [59], geo-location [60], social graph structure [43], content [28], stylometric features [3], or RNA expressions [6] In this paper, we tackle deanonymization of devices within an emerging technology, Federated Learning, which enables users to participate towards the learning of an ML model using their data in a private and anonymous manner. Moreover, we identify the inherent selection bias of individuals (e.g., capturing more photos of food) as quasiidentification signal that leads to their deanonymization.

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, 12, 16, 29, 57, 70, 73, 77, 81]), 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 [47, 56, 71], membership inference [60, 61, 65] which identifies if a particular example was used during training, attribute inference [25,52] to identify properties that holds true for subsets of data, and model inversion [24,37] 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 Distributed Machine Learning. Distributed ML on decentralized private user-generated data sources also referred to as Collaborative ML [64], or Federated Learning [13,50] – is gaining popularity is it securely enables largescale 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 especially 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 gradient information surface anonymously shared by devices to the server. The server requires raw access to these gradient signals for aggregation, opening up threats to mount inference attacks that violate users' privacy. Recently, [55] 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. The paper more similar to ours in spirit is [52], 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, while our goal can be cast as inferring a highly sensitive attribute (i.e., user identity), we also largely focus on understanding and exploiting properties of the underlying user data distribution.

3 Background, Notation and Terminology: Federated Learning

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 [13], optimization techniques [41, 50], strategies to improve communication [42], aggregation [14], implementation [1], and applications [18, 31, 76]. To keep the background in this section concise, we present key concepts to understand: (i) how devices generate model parameter deltas using the FederatedAveraging [51] algorithm; and (ii) how users communicate the parameter deltas to the server by presenting the communication protocol common in FL [13, 52, 55].

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

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{arg\,min}} H(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{arg\,min}} \frac{1}{n} \sum_{i} L(f_{\mathbf{w}}(\mathbf{x}_{i}), \, \mathbf{y}_{i}) \quad (1)$$

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

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

Federated Averaging Algorithm. Given the data \mathcal{D}_k partitioned among devices $k \in \mathbb{K}$, the objective is to learn parameters w of the model f_w , in the presence of a server S. We use the popular FederatedAveraging algorithm [49, 50] (Algorithm 1 in Appendix A) proposed specifically to perform training on non-IID and imbalanced decentralized data; this has also served as the footing for multiple prior works [14,26,51,66]. 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 $f_{\mathbf{w}}$ using the local data $\mathcal{D}_{k}^{\text{private}}$ and only communicate incremental model update $\Delta \mathbf{w}_{k}^{t}$ towards the server's global model \mathbf{w}^t . The server aggregates (such as by averaging) parameter deltas from multiple devices and shares back an updated improved model after each round. Over multiple rounds of communications, the devices converge to model parameters \mathbf{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 Federated Averaging algorithm, we expect our results to generalize to a broad class of decentralized algorithms which involve periodically exchanging model parameter deltas.

Communication Protocol. The devices in FL *anonymously* communicate [13,31,76] the local parameter deltas $\Delta \mathbf{w}_k^t$ (optionally with analytic information) to the server. Due to this protocol, a bidirectional trust between devices and server is

required. From the devices' side, that they share genuine parameters and not participate in attacks, such as poisoning [37] and backdooring [7]. Alternately, a certain amount of trust is required from the server's side as well; to ensure that the user-generated parameters are solely used for aggregation and not to e.g., perform inference attacks [52,55].

4 Deanonymization Attacks in Federated Learning

In this section, we begin by presenting our threat model to deanonymize devices within FL and the knowledge possessed by the adversary. We then discuss an insight to why this threat arises and work towards our attack models and scenarios.

4.1 Threat Model

We consider a Federated Learning (FL) setup [13,50] with $K \geq 2$ anonymous devices and a server S that periodically aggregates model parameter deltas $\Delta \mathbf{w}_k^t$. The anonymity of devices is either enforced by architectural design (default in FL) or by the user authenticating with a pseudonym. Privacy and anonymity encourages users to share their personal data for training ML models beneficial to them, especially when the training data is sensitive e.g., fitness, location.

Central to FL is learning an ML model $f_w: X \to \mathcal{Y}$ using decentralized sources of private user data. There exists an abundant amount of literature [24, 37, 61, 65] that generally identifies privacy risks of ML models trained on private data. For instance, recovering facial features [24, 37] from ML models trained on private user-specific face data. We argue a more relevant scenario, especially in a collaborative learning setup, is where individuals participate towards learning a useragnostic ML model which can be trained independent to users' private information. For instance, a general object classifier (with classes e.g., tv, flower, that aids photo organization), or a language model to improve keyboard predictions (which is in the spirit of existing FL deployed scenarios [31,76]). Our broader goal is to identify threats in such scenarios, where ML models do not require learning private (e.g., facial) features from data to perform well at test-time.

Within the setup of FL of a user-agnostic model f_w , we take the role of a malicious server S who needs to aggregate raw parameter deltas periodically communicated by anonymous devices. The role of a malicious server has received attention recently e.g., server performs membership inference [55], attribute inference [52]. There have also been incidents where this has occurred in the real-world e.g., a covert project Nautilus-S [23] which was designed to deanonymize Tor users and create a database mapping them to devices. In our threat model, we wish to exhibit that this device-to-user mapping can be similarly achieved within FL. To achieve this, we assume the adversary has access to limited prior information of the user e.g., images or text they shared on social media.

Furthermore, we analyze challenging prior distributions, such as when access to prior data is limited, and differs from users' actual distribution on device. To hint at results that will be discussed in Section 6.1.1, we find that deanonymization is possible even in these challenging scenarios, using as few as 1-10 prior data points, with the attack becoming highly effective when more prior data is gathered. The latter is alarming since many popular figures are highly active on social media, where vast amounts of prior data can be gathered.

Formally, our threat model aims to perform:

$$f^{\text{adv}}: \mathcal{D}_{u}^{\text{prior}} \times \Delta \mathbf{w}_{\text{anon}}^{t} \to u \stackrel{?}{=} \text{anon}$$
 (3)

where $\mathcal{D}_u^{\text{prior}}$ is some prior distribution of the user and $\mathbf{w}_{\text{anon}}^t = \nabla_{\mathbf{w}} f(\mathcal{D}_{\text{anon}}^{\text{private}}; \mathbf{w}^{t-1})$ is an anonymous parameter delta by device 'anon'. We always consider a challenging setting for the adversary where $\mathcal{D}^{\text{prior}} \cap \mathcal{D}_u^{\text{private}} = \emptyset \ \forall u \in \mathbb{U}$ i.e., no examples are common to both adversary's prior information set and users' private data. Furthermore, we make a mild assumption that the device 'anon' predominantly contains data of a single user. We find this reasonable since FL primarily caters to devices such as personal smartphones.

In the above formulation lies a technical challenge. We want to learn a mapping between two different modalities: an empirical distribution over users' prior data $\mathcal{D}_u^{\text{prior}} \in \mathcal{X} \times \mathcal{Y}$ and model parameters $\mathbf{w}_{\text{anon}}^t \in \mathbb{R}^D$. In Section 4.3, we will address how to overcome this.

4.2 Adversarial Knowledge

Our threat model of deanonymizing users in FL relies on an adversary with access to some prior information $\mathcal{D}_u^{\text{prior}}$ of users $u \in \mathbb{U}$. Apart from the *amount* of information that we will look at in the experimental results section, we also consider the *distribution* of this prior information w.r.t private data on the FL device $\mathcal{D}_u^{\text{private}}$. More specifically, 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., $\mathcal{D}_u^{\text{prior}}$, $\mathcal{D}_u^{\text{private}} \stackrel{\text{iid}}{\sim} \mathcal{D}_u$ This could occur with 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. In this case, 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.
- (iii) photoset **prior**: Since timestamp data of examples is unavailable in the PIPA dataset [80], we sample prior and private data non-IID by factoring in album information (photoset field). We remark that similar to chrono prior,

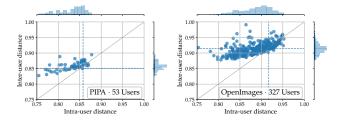


Figure 2: **Variations in user data.** Each point represents distances computed over the image set of a single user.

this is an equally challenging split as photosets often exhibit vastly different content.

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

4.3 Selection Bias and Biased Estimators

The core idea of our threat model is to use users' selection bias as a quasi-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, 22, 34] that results in subtle variations of how humans capture data. For instance, Alice's interest in culinary arts might result in more variations of food 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 [13,50]. 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).

We now present an experiment to quantify inter- and intrauser variations on two public image datasets (PIPA [80] and OpenImages [44]). 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 [39] 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 6.1.6, we will 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 (Algorithm 1) with a bias error: Bias $[\mathbf{w}_k] = \mathbb{E}[\mathbf{w}_k] - \mathbf{w}^*$, where the expectation term is over the user's training data distribution \mathcal{D}_u and \mathbf{w}^* is the optimal estimator. We conjecture (validated in §6.1.6) that the bias error signal is consistently encoded in both: (i) the parameter deltas transmitted by user's device $\Delta \mathbf{w}_u^t$; and (ii) when estimating on prior data of the user $\mathbf{w}_u^{\text{prior}} = \text{SGD}(\mathcal{D}_u^{\text{prior}})$. This leads us to reformulate the threat model (Eq. 3) in the parameter delta space:

$$f^{\mathrm{adv}}: \Delta \mathbf{w}_{u}^{\mathrm{prior}} \times \Delta \mathbf{w}_{\mathrm{anon}}^{t} \to u \stackrel{?}{=} \mathrm{anon}$$
 (4)

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

4.4 Attacks

We present two variants of deanonymization attack models: Re-identification and Matching. The former aims to classify parameter deltas to a certain user, while the latter learns a metric function in the space of parameter deltas. While the attacks we propose in this section are quite simple, they are nonetheless highly effective, as will be shown in Section 6.

Re-identification Attack. In the re-identification attack, the attacker learns an *a priori* mapping $f^{\text{re-id}}: \Delta \mathbf{w}_u^{\text{prior}} \to u$. We consider the following attack models $f^{\text{re-id}}$:

- (i) Chance: Classification performed uniformly at random.
- (ii) SVM: The adversary trains a multi-class linear Support Vector Machine (SVM) to classify users.
- (iii) **K-NN:** The adversary performs the K-Nearest Neighbor classification. Based on performance, we use K=10.
- (iv) MLP: The adversary trains a Multilayer Perceptron classifier (Fig. 13a) with a single hidden layer of 128 units and ReLU activation, using SGD with learning rate 0.01, 0.9 momentum and 10⁻⁶ LR decay.

The core idea is that $f^{\text{re-id}}$ learns user-specific biases from parameter deltas which then generalize at test time:

$$f^{\text{re-id}}: \Delta \mathbf{w}_{\text{anon}} \to u$$
 (5)

Matching Attack. The adversary's objective is to predict the match probability of a pair of distinct parameter deltas:

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

where one or both parameter deltas are anonymous. We consider the following matching attack models:

(i) Chance: The adversary predicts $p \sim \text{Unif } (0,1)$ independent of model parameter deltas.

(ii) MLP: The adversary trains the MLP-based reidentification model as before to predict per-user probabilities $(\mathbb{P}[i=u] \text{ and } \mathbb{P}[j=u])$ for every user u given the parameter deltas. We then compute the final match probability by $p = \max_{u} \mathbb{P}[i=u] \cdot \mathbb{P}[j=u]$.

(iii) Siamese: The adversary trains a Siamese network [15] with metric learning [75]. We found best performance with (architecture in Appendix B): (a) two FC-128 layers with ReLU activations which individually encodes Δw_i , Δw_j into a 128-dim embedding (b) L_1 distance layer to represent distance between these embeddings (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} .

4.5 User Scenarios

To tackle the case where set 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.

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 $\Delta w_{\rm anon}$ could map either to a seen or an unencountered user. This presents a more 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.

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.

Training Models f_w . For each dataset, we train models f_w using FederatedAveraging (Algorithm 1, Appendix A) [50] and are represented as X-FL. For reference, models (represented as X-SGD) are also trained in a centralized fashion i.e., standard training from a single pool of training data. 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 η =0.01, resulting in 1-4 GPU days to train a single model for a particular architecture, dataset and scenario.

Evaluation of models for task performance is performed on a 20% held-out test set of user data.

Experimental Setup. 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 §4.5). When setting up FL, we place half of \mathcal{D}_u on the user's anonymous device and reserve the remaining to be used as adversary's prior knowledge. All models are written in Python using the Keras [19] library with a TensorFlow [1] back-end. Training and evaluation of all models are performed on machines with Nvidia Tesla V100 GPUs with 16 GB memory.

We now present the datasets used to train and evaluate the collaboratively trained models f_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) helps us to group examples based on user and place them on devices in FL. Each of the following paragraphs discusses the (i) dataset \mathcal{D} ; (ii) corresponding task $\mathcal{X} \to \mathcal{Y}$; (iii) training model $f_w: \mathcal{X} \to \mathcal{Y}$ to perform the task; and (iv) evaluation of f_w . We perform (iii) and (iv) separately for different distributions of user data on devices (e.g., chrono, §4.2).

(i) PIPA. PIPA [80] is a large-scale dataset of \sim 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 [40] that detects 80 COCO [46] 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 $f_{\pmb{w}}: \mathbb{R}^{224 \times 224 \times 3} \to \mathbb{R}^{19},$ for this dataset in an FL setup. We use the MobileNet [39] 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. We find the model displays strong performance (see Table 2) compared to baselines.

(ii) OpenImages. OpenImages [44] 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. We evaluate the models using Average Precision (AP) scores and present the results in Table 2.

(iii) Blog Authorship. The Blog Authorship Corpus [63] is a large-scale dataset consisting of \sim 681K posts collected from

| Dataset (\mathcal{D}) | Type | Task | # Users | # Examples | Input (X) | Output (\mathcal{Y}) | Model (f _w) |
|-------------------------|----------|----------------------------|---------|------------|-------------|------------------------|-------------------------|
| PIPA [80] | Visual | Multi-label classification | 53 | 33,051 | Image | Labels | CNN-PIPA-FL |
| OpenImages [44] | Visual | Multi-label classification | 327 | 317,008 | Image | Labels | CNN-OI-FL |
| Blog [63] | Language | Language Modeling | 55 | 454,090 | Text | Text | NNLM-FL |
| Yelp [17] | Language | Sentiment Analysis | 118 | 85,615 | Text | Score | NNSA-FL |

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

| | | PIPA | | | Openl | mages |
|--------------|--------|--------|------|------------|--------|--------|
| split | rando | m phot | oset | split | random | chrono |
| CNN-PIPA-FL | 45.1 | 37 | 7.7 | CNN-OI-FL | 62.9 | 62.2 |
| CNN-PIPA-SGI | 49.7 | 40 |).7 | CNN-OI-SGD | 68.0 | 67.8 |
| K-NN | 14.9 | 15 | 5.8 | K-NN | 9.7 | 13.6 |
| Chance | 9.5 | 9 | .7 | Chance | 6.3 | 6.3 |
| | Blo | g | | | Yel | p |
| split | random | chrono | | split | random | chrono |
| NNLM-FL | 28.02 | 27.83 | | NNSA-FL | 0.716 | 0.708 |
| NNLM-SGD | 28.62 | 28.22 | | NNSA-SGD | 0.576 | 0.602 |
| Chance | 0.09 | 0.09 | | Chance | 1.472 | 1.514 |
| | | | | | | |

Table 2: **Evaluation of** f_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.

19K bloggers from 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): $P(x_t|x_{t-i},\cdots,x_{t-1}; w)$ i.e., predicting probability distribution of the next word x_t in a sequence given contextual information. Language models trained in an FL architecture are currently deployed to enable smart compose keyboards [76]. We train a Neural Network Language Model [9] using an embedding layer (with E=100 dims), LSTM layer [38] (with L=64 hidden units), and a fully-connected layer (with vocabulary size V=5000). A summary of Top5 accuracy scores are presented in Table 2.

(iv) Yelp. The Yelp Dataset [17] is a large-scale dataset consisting of \sim 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: $y = f_w([x_1, x_2, \cdots])$, where $y \in [1, 5]$ is the rating and 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. A summary of evaluation is is presented in Table 2.

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 training scenario, we now detail the training of *deanonymization attack* models (§4.4), evaluate their effectiveness, and work towards understanding how the parameter deltas leak user-identifiable information.

Evaluation Metrics. We use the following metrics (computed using scikit-learn [58]) to evaluate the adversary's attack performance:

- Mean Average Precision (AP): Adversary's precisionrecall curves for held-out user data is computed. We then compute the per-user Average Precision (area under the precision-recall curves). We report the mean of Average Precisions across users.
- 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: □×.
- 3. Top-1 accuracy: We compute the classification success rates over all parameter deltas in the test set.
- 4. Top-5 accuracy: We compute the classification success rates, where the prediction is successful if the ground-truth user is among the top 5 predictions.

These metrics are common among classification tasks e.g., [21,46,74] for AP and [20,32,45] for Top-1/5 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^{adv} . We train the ML models (f_{w} in Table 1) in an FL system simultaneously using two disjoint sets of devices per user: (a) \mathbb{K}_{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 §4.4). For simplicity, we restrict each of these sets to contain a single user. During training of f_{w} over multiple rounds, we accumulate the parameter deltas Δw_{k}^{t} communicated by all devices in FL. To train the attack models f^{adv} , we use the set of parameter deltas $\{(\Delta w_{k}^{t}, u) : k \in \mathbb{K}_{\text{prior}}\}$, where we know a priori the device k to user u mapping. We will in detail discuss training data-limited adversaries in Section 6.1.3. We evaluate attacks on the disjoint set of parameter deltas $\{\Delta (w_{k}^{t}, u) : k \in \mathbb{K}_{\text{anon}}\}$.

Representing Δw_k^t for Attacks. The parameter deltas contain hundred thousands to millions of parameters. To enable faster training and evaluation of attack models, we choose a subset of parameters by representing Δw_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 will be discussed in Section 6.1.4. Furthermore, we flatten Δw_k^t into a vector and L_2 normalize it.

6.1 Effectiveness of Deanonymization Attacks

In this section, we evaluate the effectiveness of deanonymization attacks across a range of scenarios and explore various factors that influence the attack effectiveness. In addition, we also present experiments to understand the effectiveness.

6.1.1 Impact of Prior Distributions on Attack Models

We evaluate our attacks when performing re-identification and matching (§4.4) attacks for choices of prior distributions (§4.2) and datasets (§5).

The adversary's goal is to re-Re-identification Attack. identify the user using the attack model $f^{\text{re-id}}: \Delta w_{\text{ukn}}^t \to u$. From results 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 \times on PIPA vs. 175 \times on OpenImages under the same random prior); (v) The attack is highly 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 datadistribution among devices, which is natural in FL since users participate with personal data exhibiting unique biases (§4.3). 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 deltas contain ample user information in an FL setup that allows for effective deanonymization.



Figure 3: **profile prior.** Devices can be isolated using proxy distributions of certain profiles, such as of handguns and guitars. Rows denote private data $\mathcal{D}_u^{\text{private}}$ of users on devices.

Matching Attack. We study the matching attacks that performs the task: $f^{\text{adv}}: (\Delta w_i, \Delta w_j) \rightarrow i \stackrel{?}{=} j$. I.e., were a pair of parameter deltas generated by the same user? Table 4 presents the AP performances for matching when using one of the parameter deltas generated from the prior distribution. We find that the adversary can match the users very well. On PIPA, the MLP based attack achieves nearly perfect (99.5% AP) matching on random prior. Across all datasets and prior distributions considered, matching can be done very successfully, all greater than 79.3% achieved in the Yelp-chrono split.

Attacking using uprofile prior. In the previous attacks we looked at the task of deanonymizing devices by associating the parameter deltas to prior distributions of users. We now look at the 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 3 we display the top OpenImages users, 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.

6.1.2 Impact of Number of Seen and Unseen Users

We now consider the open-world scenario (discussed in Section 4.5), where at test time the adversary encounters a new set of unseen users when training the attack model.

User Split. We split the users \mathbb{U} into three variably-sized disjoint sets: (a) \mathbb{U}_{unseen} : prior data is unavailable and should be classified as unseen at test-time; (b) \mathbb{U}_{seen} : prior data is

| | | PIPA (#Users $U = 53$) | | | | | | (| OpenImage | S(U = 327) | | |
|--------|------------|-------------------------|-------|--------------------|-------|---------------|-------------------|-------|-----------|--------------------|-------|-------|
| | random | | ph | otoset | | random chrono | | | rono | | | |
| | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 | AP | Top-1 | Top-5 |
| MLP | 91 (48×) | 84.7 | 96.3 | 42.2 (22×) | 40 | 68.8 | 53.7 (175×) | 51.9 | 77.9 | 32.5 (106×) | 31.9 | 57.1 |
| SVM | 81.3 (43×) | 89.3 | 91.9 | $27.7 (15 \times)$ | 43.7 | 49.6 | $49 (159 \times)$ | 66.5 | 67 | $24.6 (80 \times)$ | 41.7 | 42.5 |
| kNN | 85.4 (45×) | 82.6 | 92.6 | 31.5 (17×) | 38.4 | 54.8 | 46 (150×) | 49.2 | 63.9 | 25.1 (82×) | 30.3 | 43.1 |
| Chance | 1.9 (1×) | 2 | 9.9 | 1.9 (1×) | 2 | 9.9 | 0.3 (1×) | 0.3 | 1.5 | $0.3(1 \times)$ | 0.3 | 1.5 |

| | | Blog (<i>U</i> = 55) | | | | | | | Yelp (U | = 118) | | |
|--------|------------|-----------------------|-------|--------------------|-------|-------------|------------|-------|---------|--------------------|-------|-------|
| | random | | c. | hrono | | random chro | | | rono | ono | | |
| | 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 | $27.0 (15 \times)$ | 42.1 | 46.0 | 25.9 (31×) | 43.2 | 44.9 | $17.1 (20 \times)$ | 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** ($\Delta w_{anon} \rightarrow u$). Performed in a closed-world. Chance-level AP $\approx 1/U$.

| | P | IPA | Open | Images |
|--------|--------|----------|--------|--------|
| - | random | photoset | random | chrono |
| MLP | 99.5 | 91.2 | 98.2 | 94.8 |
| Chance | 49.1 | 49.4 | 50.8 | 49 |
| | В | log | Yel | lp |
| | random | chrono | random | chrono |
| MLP | 95.3 | 91.9 | 83.4 | 79.3 |
| Chance | 50.0 | 48.7 | 48.9 | 50.7 |

Table 4: **Matching Attack Evaluation** $((\Delta w_i, \Delta w_j) \rightarrow i \stackrel{?}{=} j)$. Performed in a closed-world setup. Chance AP $\approx 50\%$.

available and should be deanonymized at test-time; and (c) $\mathbb{U}_{holdout}$: these users are reserved purely for training purposes.

Re-identification Setup. Previously in the closed-world scenario, we trained the MLP (§4.4) classifier $f^{\text{re-id}}: \Delta \mathbf{w}_k \to u$ 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 deltas to train the unseen class.

Matching Setup. We train a Siamese network (§4.4) using parameter deltas from held-out and seen set of users. Given a pair $(\Delta w_i, \Delta w_j)$, the network predicts the probability $\mathbb{P}[i=j]$ of being generated by the same user.

Evaluation. The performances are evaluated at different 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 most challenging prior distributions per dataset are presented in Figure 4. 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, due to increase

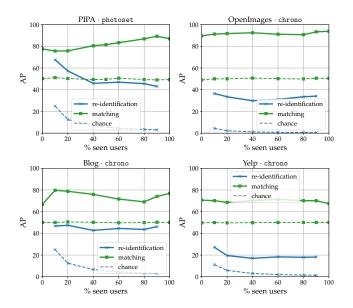


Figure 4: **Open-world evaluation.** Across re-identification (MLP) and matching (Siamese) attack models.

in output-space of the attack as %seen users increase, we notice a slight drop in AP performance (67% \rightarrow 43% in PIPA). However, performance compared to chance-level significantly increases (3× \rightarrow 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). Even in the presence of unseen users at test time, our re-identification and matching attacks are robust and generalizable.

6.1.3 Amount of Training Data

We now study the influence of data-limitation in deanonymization attacks in a closed-world re-identification scenario. We

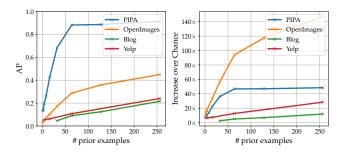


Figure 5: **Number of prior examples per user.** Evaluated on closed-world re-identification.

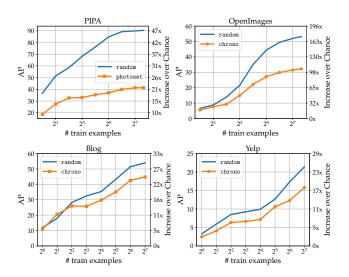


Figure 6: **Number of training examples per user.** Evaluated on closed-world re-identification.

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 5, 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 ($7 \times$) 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 $16 \rightarrow 32$ 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 $5 \times$ 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 for $\{\Delta w_k^t : k \in \mathbb{K}_{anon}\}$ attack models. We train multiple re-identification MLP adversary models, each trained on a random subset of training data with increasing sizes. In Figure 6, we observe an adversary can train reasonably effective attack models, even with extremely limited labeled data. In particular, attack

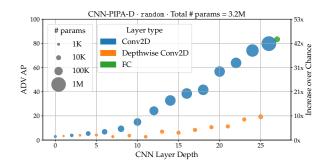


Figure 7: **Re-identification performance by depth.** Bubble sizes indicate the number of parameters in each layer. Last two layers contains 1M and 19K parameters respectively.

| | | NNLM-I | O (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.

performance of $3\times -22\times$ 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.

6.1.4 Impact of Parameter Layers

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 [79] – 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. We train and evaluate re-identification MLP models on random prior using parameter deltas contributed by each individual trainable layer. This results in 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 perform the experiment on CNN-OI-FL as it would require > 3TB.

From layer-wise performances in Figure 7 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 informative; (iii) in the RNN-based models, the LSTM parameters are more informative for language modeling, whereas it is the embedding layer for sentiment analysis.

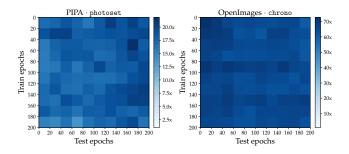


Figure 8: **Effect of the epoch** t. On the re-identification attack $\Delta \mathbf{w}_{\text{anon}}^t \rightarrow u$. As an example, the top-right cell denotes when the MLP was trained on $\Delta \mathbf{w}_u^t, t \in [0, 20]$ and evaluated on $\Delta \mathbf{w}_{\text{anon}}^t, t' \in [180, 200]$

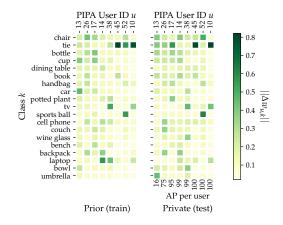


Figure 9: User bias visualized on parameter deltas.

6.1.5 Impact of Optimization State

We now analyze the influence of training progress of the ML model on deanonymization attacks. We group the parameter deltas (separately for train and test attack sets), based on the epoch ranges during which they were generated. We split parameter deltas collected during training of f_w over 200 epochs into 10 ranges, each with 20 epochs. We train and evaluate the MLP re-identification attack model over all 10×10 train-eval pairs. From Figure 8, 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.

6.1.6 Reasoning About Effectiveness of Attacks

In Section 4.3 (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 deltas, even when they are computed on different (prior and private) sets of users' data. To validate, we take a closer look at the parameter deltas $\Delta \mathbf{w}_u^{\text{prior}}$, $\Delta \mathbf{w}_u^{\text{private}} \in \mathbb{R}^{D \times K}$ in the FC layer of eight users in the

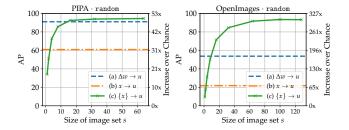


Figure 10: **Effect of aggregation of user data.** Evaluated on closed-world re-identification attack.

PIPA FL setup, where K (=19) is the number of classes and D(=1024) represents weights per class. In Figure 9, we illustrate bias per user individually for (a) and (b) in the parameter delta space by computing the L_2 -norm of each of the K class weight vectors (columns in Δw_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.

6.1.7 Comparison with Deanonymizing Images

We have thus far targeted the bias signal in the *parameter delta space* $\Delta \mathbf{w} \in \mathbb{R}^D$ to perform deanonymization attacks: $f_{\mathbf{w}}^{\text{adv}}: \Delta \mathbf{w}_{u}^{\text{prior}} \times \Delta \mathbf{w}_{\text{anon}}^t \to u \stackrel{?}{=} \text{anon.}$ We now target the *data space* $\mathbf{x} \in X$ and ask how this compares to a hypothetical deanonymization attack directly on the users' private data: $f_{\mathbf{x}}^{\text{adv}}: \mathbf{x}_{u}^{\text{prior}} \times \mathbf{x}_{\text{anon}} \to u \stackrel{?}{=} \text{anon.}$ The experiment will help us understand the information gain (or loss) for deanonymization attacks resulting from the encoding the inputs to parameter deltas by training the ML model.

We study this using three deanonymization attack models with different input types (parameters, image, image set), but all predicting the user *u*:

- (a) Model update re-identification ($u = f^{\text{adv}}(\Delta w_{\text{anon}})$): We reuse the MLP re-identification attack (§4.4, Eq. 5).
- (b) **Image re-identification** $(u = f^{\text{adv}}(\mathbf{x}))$: We train a Mobilenet CNN on the prior data $\mathcal{D}^{\text{prior}} = \{(\mathbf{x}_i, u)\}.$
- (c) Image set re-identification $(u = f^{adv}(\{x\}))$: We classify variably-sized *sets* of images, where each set is now represented by computing the mean of individual features $\phi(x_i)$ extracted from (b).

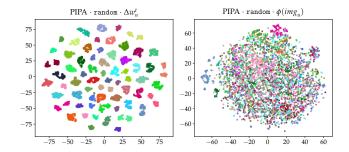


Figure 11: **t-SNE visualization**. Of parameter deltas Δw_i (left) and inputs features x_i (right). Colors indicate users.

From Figure 10 when comparing (a) and (b), we observe that re-identification using a single update Δw_{anon} (blue dashed line) is surprisingly more effective $(1.5-2.5\times)$ than deanonymizing using raw data \mathbf{x}_{anon} (orange dash-dotted line). Moreover, in Figure 11 we visualize how well the attack input spaces to both these models are clustered using the t-SNE [48] algorithm. t-SNE projects high-dimension data (in our case, >19k dimensions) onto a 2-dimensional plane while approximately preserving the distance graph, making the data visualization-friendly. We observe significantly lower variances among inputs in the parameter space (Fig. 11 left) compared to raw data (Fig. 11 right) and hence leading to better attacks in the parameter space. However, upon evaluating (c) whose input is a set of user examples, we find (green lines in Fig. 10) that larger sizes of inputs results in better representations of the users' distribution and corresponding biases, which leads to better deanonymization attacks.

We find aggregate statistics of users – which is a property of both the parameter deltas Δw generated during FL and the raw user data set – provides a powerful signal to perform deanonymization attacks. Surprisingly in some cases (e.g., PIPA), we additionally find the user-bias signal for deanonymization attacks is equally strong in both the parameter deltas and the raw user data itself.

7 Countermeasures

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

We attributed (§6.1.6) the effectiveness of the attacks to user bias, which is a powerful statistical signal in both the limited set of prior data (that the adversary possesses) and the users' private data on device anonymously participating in FL. The focus of our mitigation strategies is to perturb the data bias on the anonymous device, in order to provide a false signal to the adversary. We spell out our requirements for the defense as: (a) not decrease utility (performance of f_w) too much; (b) involve low computation overhead; (c) not rely on a trusted-third party; and (d) allow users to selectively

| \mathcal{D} | Source (\mathcal{D}) | $\mathcal{D}^{	ext{bkg}}$ | Source (\mathcal{D}^{bkg}) | $ \mathcal{D}^{\mathrm{bkg}} $ |
|------------------------------|------------------------|---------------------------|------------------------------|--------------------------------|
| PIPA [80] OpenImages [44] | Flickr Flickr | OpenImages OpenImages | Flickr Flickr | 59K 490K |
| Blog [63] | Blogger | WikiReading [36] | Wikipedia | 3M |
| Yelp [17] | Yelp | Amazon Reviews [33] | Amazon | 1.7M |

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

employ the strategy to various extents depending on personal preferences.

We assume our mitigation strategies are: (1) device-sided; (2) used by the users' anonymous device throughout the training process ($t \ge 0$); (3) aimed to prevent deanonymization attacks in Section 4.4:

7.1 Methods

Based on the requirements and assumptions, we propose datacentric mitigation strategies: devices adversarially bias their data distributions on devices, rather than directly perturb model parameters. More specifically, users mix their original data \mathcal{D}_u with certain "background" data \mathcal{D}^{bkg} to "blend into the crowd", thereby rendering the parameters less userspecific. The mixing takes place before the adversary's training process. We will explain the strategies in greater detail, and discuss how they address the requirements.

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

We now look at four mitigation strategies.

Random Perturbations (noise). As a baseline, we randomly perturb parameter deltas by adding zero-centered Gaussian noises:

$$\Delta \hat{\mathbf{w}}_u \leftarrow \Delta \mathbf{w}_u + \mathcal{N}(\mu = 0, \sigma^2) \tag{7}$$

Note that the noised parameters $\Delta \hat{w}_u$ will no longer be true gradients; it is hard to predict the optimization results.

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}^{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 [30, 67])

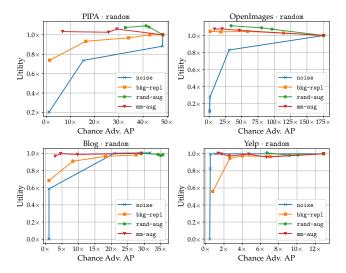


Figure 12: **Mitigation strategies evaluation.** Reidentification AP obtained by varying α and σ^2 in closed-world scenario. Top-left is the ideal region. Higher α and σ^2 values pushes operating points towards the left (i.e., lower deanonymization performance).

from \mathcal{D}^{bkg} :

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

where $\alpha \geq 0$ determines the size of augmentation. As $\alpha \to \infty$, devices' data distributions converge to \mathcal{D}^{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}^{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 culinary arts before participating in FL.

We perform this by first clustering \mathcal{D}^{bkg} into M clusters $\bigcup_{m=1}^{M} \mathcal{D}_{m}^{bkg}$. 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:

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

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

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 Section 6.1.1, we measure the adversary's performance as increase over chance-level AP scores. We measure utility by performance scores normalized to have utility=1.0 when no mitigation takes place.

For each mitigation strategy, multiple hyperparameters are considered. For noise, we consider Gaussian noise with $\mu=0$ and $\sigma^2\in\{10^{-2},10^{-1},10^0,10^1,10^2\}$. 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 $\alpha\in\{0.0,0.5,1.0,2.0\}$.

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 noise baseline in most cases decreases utility severely 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 noise. However, due to a domain-shift between \mathcal{D}^{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 [30,67]. 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 [39,62].

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 deltas 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.

Acknowledgement. This research was partially supported by the German Research Foundation (DFG CRC 1223).

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. *arXiv preprint arXiv:1807.00459*, 2018.
- [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] 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.
- [13] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046, 2019.
- [14] 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.
- [15] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *NeurIPS*, 1994.
- [16] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE S&P*, 2017.
- [17] Yelp Dataset Challenge. Yelp dataset challenge, 2013.
- [18] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.
- [19] François Chollet et al. Keras. https://keras.io, 2015.
- [20] 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.
- [21] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.
- [22] Barbara Fadem. *Behavioral science in medicine*. Lippincott Williams & Wilkins, 2012.
- [23] Jon Fingas. Hackers broke into a contractor for russia's spy agency. https://www.engadget.com/2019/07/21/hackers-break-into-russia-fsb-contractor/, 2019. Accessed August 29, 2019.
- [24] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 2015.
- [25] 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.

- [26] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. In *NIPS PPML Workshop*, 2017.
- [27] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016.
- [28] 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.
- [29] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, and Gang Wang abd Xinyu Xing. LEMNA: Explaining Deep Learning based Security Applications. In *CCS*, 2018.
- [30] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.
- [31] 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.
- [32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [33] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, 2016.
- [34] Miguel A Hernán, Sonia Hernández-Díaz, and James M Robins. A structural approach to selection bias. *Epidemiology*, 15(5):615–625, 2004.
- [35] Nestor Hernandez, Mizanur Rahman, Ruben Recabarren, and Bogdan Carbunar. Fraud de-anonymization for fun and profit. In *CCS*, 2018.
- [36] 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.
- [37] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *CCS*, 2017.
- [38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

- [39] 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.
- [40] 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.
- [41] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv* preprint arXiv:1610.02527, 2016.
- [42] Jakub Konečný, 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.
- [43] Nitish Korula and Silvio Lattanzi. An efficient reconciliation algorithm for social networks. *VLDB*, 2014.
- [44] 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 multiclass image classification. Dataset available from https://storage.googleapis.com/openimages/web/index.html, 2017
- [45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.
- [46] 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.
- [47] Daniel Lowd and Christopher Meek. Adversarial learning. In KDD, 2005.
- [48] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.
- [49] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized data. https://research.googleblog.com/2017/ 04/federated-learning-collaborative.html, 2017. Accessed January 21, 2018.

- [50] 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.
- [51] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.
- [52] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Inference attacks against collaborative learning. In *S&P*, 2019.
- [53] Arvind Narayanan and Vitaly Shmatikov. Robust deanonymization of large sparse datasets. In *S&P*, 2008.
- [54] Arvind Narayanan and Vitaly Shmatikov. Deanonymizing social networks. In *S&P*, 2009.
- [55] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Standalone and federated learning under passive and active white-box inference attacks. In *S&P*, 2019.
- [56] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In CVPR, 2019.
- [57] 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.
- [58] 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.
- [59] Daniele Perito, Claude Castelluccia, Mohamed Ali Kaafar, and Pere Manils. How unique and traceable are usernames? In *PETS*, 2011.
- [60] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *NDSS*, 2018.
- [61] 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.
- [62] 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.

- [63] Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. Effects of age and gender on blogging. In *AAAI*, 2006.
- [64] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *CCS*, 2015.
- [65] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *S&P*, 2017.
- [66] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *NeurIPS*, 2017.
- [67] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.
- [68] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. In *AMIA*, 1997.
- [69] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 1997.
- [70] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.
- [71] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.
- [72] Daisuke Wakabayashi. Google accused of inappropriate access to medical data in potential class-action lawsuit. https://www.nytimes.com/2019/06/26/technology/google-university-chicago-data-sharing-lawsuit.html, 2019. Accessed August 29, 2019.
- [73] 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.
- [74] 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.
- [75] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *NeurIPS*, 2006.
- [76] 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.
- [77] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y. Zhao. Automated Crowdturfing Attacks and Defenses in Online Review Systems. In CCS, 2017.
- [78] Ryo Yonetani, Vishnu Naresh Boddeti, Kris M Kitani, and Yoichi Sato. Privacy-preserving visual learning using doubly permuted homomorphic encryption. In *ICCV*, 2017.
- [79] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.
- [80] Ning Zhang, Manohar Paluri, Yaniv Taigman, Rob Fergus, and Lubomir Bourdev. Beyond frontal faces: Improving person recognition using multiple cues. In *CVPR*, 2015.
- [81] Xiao Zhang and David Evans. Cost-Sensitive Robustness against Adversarial Examples. In *ICLR*, 2019.

Appendices

A Federated Averaging Algorithm

```
Algorithm 1: FederatedAveraging [50] 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 \text{sample } M \text{ devices from } \mathbb{K}
       for client k \in \mathbb{K}_t do
             \Delta \mathbf{w}_{\nu}^{t+1} \leftarrow \text{DeviceUpdate}(k, \mathbf{w}^{t})
       end
       \mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \sum_{k \in \mathbb{K}_t} \frac{n_k}{n} \Delta \mathbf{w}_k^{t+1}
 end
DeviceUpdate(k, \mathbf{w}^t):
 \mathcal{B} \leftarrow \text{split local data } \mathcal{D}_{\iota}^{\text{private}} \text{ into batches of size } B
 for local epoch i \leftarrow 1 to E do
       for batch b \in \mathcal{B} do
             \mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(f_{\mathbf{w}}; b)
       end
 end
 \Delta \mathbf{w} \leftarrow \mathbf{w}^t - \mathbf{w}
 return \Delta w
```

B Supplemental Details on Attack Models

We present the architecture of the MLP and Siamese networks discussed in 4.4 in Figures 13a and 13b respectively.

C Dataset Examples

We present example data from our datasets discussed in Section 5. The examples are indicated with the label of the (ob-

fuscated) real-world user and the corresponding timestamps.

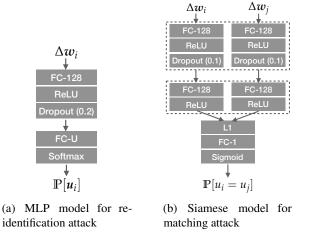


Figure 13: Architectures of linkability attack models. Dotted lines indicate shared layers.



Figure 14: Examples from our datasets: (top) OpenImages and (bottom) Yelp. Images here are grouped by the anonymized userid and captured/review date.