Exploiting Unintended Feature Leakage in Collaborative Learning*

Luca Melis†
UCL
luca.melis.14@alumni.ucl.ac.uk

Congzheng Song†
Cornell University
cs2296@cornell.edu

Emiliano De Cristofaro
UCL & Alan Turing Institute
e.decristofaro@ucl.ac.uk

Vitaly Shmatikov
Cornell Tech
shmat@cs.cornell.edu

Abstract

Collaborative machine learning and related techniques such as federated learning allow multiple participants, each with his own training dataset, to build a joint model by training locally and periodically exchanging model updates. We demonstrate that these updates leak *unintended* information about participants' training data and develop passive and active inference attacks to exploit this leakage. First, we show that an adversarial participant can infer the presence of exact data points—for example, specific locations—in others' training data (i.e., *membership* inference). Then, we show how this adversary can infer *properties* that hold only for a subset of the training data and are independent of the properties that the joint model aims to capture. For example, he can infer when a specific person first appears in the photos used to train a binary gender classifier. We evaluate our attacks on a variety of tasks, datasets, and learning configurations, analyze their limitations, and discuss possible defenses.

1 Introduction

Collaborative machine learning (ML) has recently emerged as an alternative to conventional ML methodologies where all training data is pooled and the model is trained on this joint pool. It allows two or more participants, each with his own training dataset, to construct a joint model. Each participant trains a local model on his own data and periodically exchanges model parameters, updates to these parameters, or partially constructed models with the other participants.

Several architectures have been proposed for distributed, collaborative, and federated learning [9, 11, 33, 38, 62, 68]: with and without a central server, with different ways of aggregating models, etc. The main goal is to improve the training speed and reduce overheads, but protecting privacy of the participants' training data is also an important motivation for several recently proposed techniques [35, 52]. Because the training data never leave the participants' machines, collaborative learning may be a good match for the scenarios where this data is sensitive (e.g., health-care records, private images, personally identifiable information, etc.). Compelling applications include training of predictive keyboards on character sequences that users type on their smartphones [35], or using data from multiple hospitals to develop predictive models for patient survival [29] and side effects of medical treatments [30].

Collaborative training, however, *does* disclose information via model updates that are based on the training data. The key question we investigate in this paper is: **what can be inferred about a participant's training dataset from the model updates** revealed during collaborative model training?

Of course, the purpose of ML is to discover new information about the data. Any useful ML model reveals something about the population from which the training data was drawn. For example, in addition to accurately classifying its inputs, a classifier model may reveal the features that characterize a given class or help construct data points that belong to this class. In this paper, we focus on inferring "unintended" features, i.e., properties that hold for certain subsets of the training data, but not generically for all class members.

The basic privacy violation in this setting is *membership inference*: given an exact data point, determine if it was used to train the model. Prior work described passive and active membership inference attacks against ML models [24, 53], but collaborative learning presents interesting new avenues for such inferences. For example, we show that an adversarial participant can infer whether a specific location profile was used to train a gender classifier on the FourSquare location dataset [64] with 0.99 precision and perfect recall.

We then investigate passive and active *property inference* attacks that allow an adversarial participant in collaborative learning to infer properties of other participants' training data that are not true of the class as a whole, or even *independent* of the features that characterize the classes of the joint model. We also study variations such as inferring when a property appears and disappears in the data during training—for example, identifying when a certain person first appears in the photos used to train a generic gender classifier.

For a variety of datasets and ML tasks, we demonstrate successful inference attacks against two-party and multi-party collaborative learning based on [52] and multi-party federated learning based on [35]. For example, when the model is trained on the LFW dataset [28] to recognize gender or race, we infer whether people in the training photos wear glasses—a property that is *uncorrelated* with the main task. By contrast, prior property inference attacks [2, 25] infer only properties that characterize an entire class. We discuss this critical distinction in detail in Section 3.

Our key observation, concretely illustrated by our experiments, is that modern deep-learning models come up with separate internal representations of all kinds of features, some of which are independent of the task being learned. These "unintended" features leak information about participants' training

*In Proceedings of 40th IEEE Symposium on Security & Privacy (S&P 2019).
†Authors contributed equally.

data. We also demonstrate that an *active* adversary can use multi-task learning to trick the joint model into learning a better internal separation of the features that are of interest to him and thus extract even more information.

Some of our inference attacks have direct privacy implications. For example, when training a binary gender classifier on the FaceScrub [40] dataset, we infer with high accuracy (0.9 AUC score) that a certain person appears in a single training batch even if half of the photos in the batch depict other people. When training a generic sentiment analysis model on Yelp healthcare-related reviews, we infer the specialty of the doctor being reviewed with perfect accuracy. On another set of Yelp reviews, we identify the author even if their reviews account for less than a third of the batch.

We also measure the performance of our attacks vis-à-vis the number of participants (see Section 7). On the image-classification tasks, AUC degrades once the number of participants exceeds a dozen or so. On sentiment-analysis tasks with Yelp reviews, AUC of author identification remains high for many authors even with 30 participants.

Federated learning with model averaging [35] does not reveal individual gradient updates, greatly reducing the information available to the adversary. We demonstrate successful attacks even in this setting, e.g., inferring that photos of a certain person appear in the training data.

Finally, we evaluate possible defenses—sharing fewer gradients, reducing the dimensionality of the input space, dropout—and find that they do not effectively thwart our attacks. We also attempt to use participant-level different privacy [36], which, however, is geared to work with thousands of users, and the joint model fails to converge in our setting.

2 Background

2.1 Machine learning (ML)

An ML model is a function $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ parameterized by a set of *parameters* θ, where $\mathcal{X}$ denotes the input (or feature) space, and $\mathcal{Y}$ the output space.

In this paper, we focus on the supervised learning of classification tasks. The *training data* is a set of data points labeled with their correct classes. We work with models that take as input images or text (i.e., sequences of words) and output a class label. To find the optimal set of parameters that fits the training data, the training algorithm optimizes the *objective (loss) function*, which penalizes the model when it outputs a wrong label on a data point. We use $L(x, y; \theta)$ to denote the loss computed on a data point (x, y) given the model parameters θ, and $L(b; \theta)$ to denote the average loss computed on a batch b of data points.

Stochastic Gradient Descent (SGD). There are many methods to optimize the objective function. Stochastic gradient descent (SGD) and its variants are commonly used to train artificial neural networks, but our inference methodology is not specific to SGD. SGD is an iterative method where at each step the optimizer receives a small batch of the training data and updates the model parameters θ according to the direction of the negative gradient of the objective function with respect to θ and scaled by the learning rate η. Training finishes when the model has converged to a local minimum, where the gradient is close to zero. The trained model is tested using held-out data, which was not used during training. A standard metric is *test accuracy*, i.e., the percentage of held-out data points that are classified correctly.

Algorithm 1 Parameter server with synchronized SGD

Server executes:
Initialize θ_0
for $t = 1$ to T **do**
  for each client k **do**
    $g_t^k \leftarrow$ **ClientUpdate**(θ_{t-1})
  end for
  $\theta_t \leftarrow \theta_{t-1} - \eta \sum_k g_t^k$ ▷ synchronized gradient updates
end for

ClientUpdate(θ):
Select batch b from client's data
return local gradients $\nabla L(b; \theta)$

Hyperparameters. Most modern ML algorithms have a set of tunable hyperparameters, distinct from the model parameters. They control the number of training iterations, the ratio of the regularization term in the loss function (its purpose is to prevent overfitting, i.e., a modeling error that occurs when a function is too closely fitted to a limited set of data points), the size of the training batches, etc.

Deep learning (DL). A family of ML models known as deep learning recently became very popular for many ML tasks, especially related to computer vision and image recognition [32, 51]. DL models are made of layers of non-linear mappings from input to intermediate hidden states and then to output. Each connection between layers has a floating-point weight matrix as parameters. These weights are updated during training. The topology of the connections between layers is task-dependent and important for the accuracy of the model.

2.2 Collaborative learning

Training a deep neural network on a large dataset can be time- and resource-consuming. A common scaling approach is to partition the training dataset, concurrently train separate models on each subset, and exchange parameters via a parameter server [9, 11]. During training, each local model pulls the parameters from this server, calculates the updates based on its current batch of training data, then pushes these updates back to the server, which updates the global parameters.

Collaborative learning may also involve participants who want to hide their training data from each other. We review two architectures for privacy-preserving collaborative learning based on, respectively, [52] and [35].

Collaborative learning with synchronized gradient updates. Algorithm 1 shows collaborative learning with synchronized gradient updates [52]. In every iteration of training, each participant downloads the global model from the parameter server, locally computes gradient updates based

Algorithm 2 Federated learning with model averaging

Server executes:
Initialize θ_0
$m \leftarrow max(C \cdot K, 1)$
for $t = 1$ to T **do**
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **do**
$\theta_t^k \leftarrow$**ClientUpdate**(θ_{t-1})
end for
$\theta_t \leftarrow \sum_k \frac{n^k}{n} \theta_t^k$ ▷ averaging local models
end for

ClientUpdate(θ):
for each local iteration **do**
for each batch b in client's split **do**
$\theta \leftarrow \theta - \eta \nabla L(b; \theta)$
end for
end for
return local model θ

on one batch of his training data, and sends the updates to the server. The server waits for the gradient updates from all participants and then applies the aggregated updates to the global model using stochastic gradient descent (SGD).

In [52], each client may share only a fraction of his gradients. We evaluate if this mitigates our attacks in Section 8.1. Furthermore, [52] suggests differential privacy to protect gradient updates. We do not include differential privacy in our experiments. By definition, record-level differential privacy bounds the success of membership inference, but does not prevent property inference that applies to a group of training records. Participant-level differential privacy, on the other hand, bounds the success of all attacks considered in this paper, but we are not aware of any participant-level differential privacy mechanism that enables collaborative learning of an accurate model with a small number of participants. We discuss this further in Section 8.4.

Federated learning with model averaging. Algorithm 2 shows the federated learning algorithm [35]. We set C, the fraction of the participants who update the model in each round, to 1 (i.e., the server takes updates from all participants), to simplify our experiments and because we ignore the efficiency of the learning protocol when analyzing the leakage.

In each round, the k-th participant locally takes several steps of SGD on the current model using his entire training dataset of size n^k (i.e., the globally visible updates are based not on batches but on participants' entire datasets). In Algorithm 2, n is the total size of the training data, i.e., the sum of all n^k. Each participant submits the resulting model to the server, which computes a weighted average. The server evaluates the resulting joint model on a held-out dataset and stops training when performance stops improving.

The convergence rate of both collaborative learning approaches heavily depends on the learning task and the hyperparameters (e.g., number of participants and batch size).

3 Reasoning about Privacy in Machine Learning

If a machine learning (ML) model is useful, it must reveal information about the data on which it was trained [13]. To argue that the training process and/or the resulting model violate "privacy," it is not enough to show that the adversary learns something new about the training inputs. At the very least, the adversary must learn *more* about the training inputs than about other members of their respective classes. To position our contributions in the context of related work (surveyed in Section 10) and motivate the need to study unintended feature leakage, we discuss several types of adversarial inference previously considered in the research literature.

3.1 Inferring class representatives

Given black-box access to a classifier model, *model inversion* attacks [16] infer features that characterize each class, making it possible to construct representatives of these classes.

In the special case—and only in this special case—where all class members are similar, the results of model inversion are similar to the training data. For example, in a facial recognition model where each class corresponds to a single individual, all class members depict the same person. Therefore, the outputs of model inversion are visually similar to any image of that person, including the training photos. If the class members are *not* all visually similar, the results of model inversion do not look like the training data [53].

If the adversary actively participates in training the model (as in the collaborative and federated learning scenarios considered in this paper), he can use GANs [22] to construct class representatives, as done by Hitaj et al. [25]. Only in the special case where all class members are similar, GAN-constructed representatives are similar to the training data. For example, all handwritten images of the digit '9' are visually similar. Therefore, a GAN-constructed image for the '9' class looks similar to *any* image of digit 9, including the training images. In a facial recognition model, too, all class members depict the same person. Hence, a GAN-constructed face looks similar to any image of that person, including the training photos.

Note that *neither technique reconstructs actual training inputs.* In fact, there is no evidence that GANs, as used in [25], can even distinguish between a training input and a random member of the same class.

Data points produced by model inversion and GANs are similar to the training inputs only if all class members are similar, as is the case for MNIST (the dataset of handwritten digits used in [25]) and facial recognition. This simply shows that ML works as it should. A trained classifier reveals the input features characteristic of each class, thus enabling the adversary to sample from the class population. For instance, Figure 1 shows GAN-constructed images for the gender classification task on the LFW dataset, which we use in our experiments (see Section 6). These images show a generic female face, but there is no way to tell from them whether an image of a *specific* female was used in training or not.

Finally, the active attack in [25] works by overfitting the

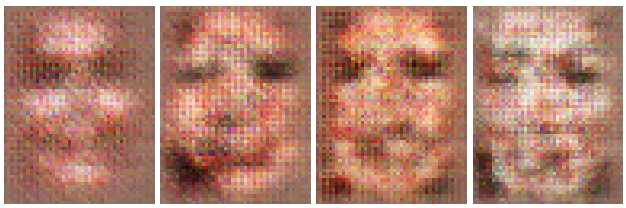


Figure 1: Samples from a GAN attack on a gender classification model where the class is "female."

joint model's representation of a class to a single participant's training data. This assumes that the entire training corpus for a given class belongs to that participant. We are not aware of any deployment scenario for collaborative learning where this is the case. By contrast, we focus on a more realistic scenario where the training data for each class are distributed across multiple participants, although there may be significant differences between their datasets.

3.2 Inferring membership in training data

The (arguably) simplest privacy breach is, given a model and an exact data point, inferring whether this point was used to train the model or not. Membership inference attacks against aggregate statistics are well-known [14, 27, 50], and recent work demonstrated black-box membership inference against ML models [24, 34, 53, 58], as discussed in Section 10.

The ability of an adversary to infer the presence of a specific data point in a training dataset constitutes an immediate privacy threat if the dataset is in itself sensitive. For example, if a model was trained on the records of patients with a certain disease, learning that an individual's record was among them directly affects his or her privacy. Membership inference can also help demonstrate inappropriate uses of data (e.g., using health-care records to train ML models for unauthorized purposes [4]), enforce individual rights such as the "right to be forgotten," and/or detect violations of data-protection regulations such as the GDPR [19]. Collaborative learning presents interesting new avenues for such inferences.

3.3 Inferring properties of training data

In collaborative and federated learning, participants' training data may not be identically distributed. Federated learning is explicitly designed to take advantage of the fact that participants may have private training data that are different from the publicly available data for the same class [35].

Prior work [2, 16, 25] aimed to infer properties that characterize an entire class: for example, given a face recognition model where one of the classes is Bob, infer what Bob looks like (e.g., Bob wears glasses). It is not clear that hiding this information in a good classifier is possible or desirable.

By contrast, we aim to infer *properties that are true of a subset of the training inputs but not of the class as a whole*. For instance, when Bob's photos are used to train a gender classifier, we infer that Alice appears in some of the photos. We especially focus on the properties that are *independent* of the class's characteristic features. In contrast to the face recognition example, where "Bob wears glasses" is a characteristic feature of an entire class, in our gender classifier study we infer whether people in Bob's photos wear glasses—even though wearing glasses has no correlation with gender. There is no legitimate reason for a model to leak this information; it is purely an artifact of the learning process.

A participant's contribution to each iteration of collaborative learning is based on a batch of his training data. We infer *single-batch properties*, i.e., detect that the data in a given batch has the property but other batches do not. We also infer *when a property appears in the training data*. This has serious privacy implications. For instance, we can infer when a certain person starts appearing in a participant's photos or when the participant starts visiting a certain type of doctors. Finally, we infer properties that characterize a participant's entire dataset (but not the entire class), e.g., authorship of the texts used to train a sentiment-analysis model.

4 Inference Attacks

4.1 Threat model

We assume that K participants (where $K \geq 2$) jointly train an ML model using one of the collaborative learning algorithms described in Section 2.2. One of the participants is the *adversary*. His goal is to infer information about the training data of another, *target participant* by analyzing periodic updates to the joint model during training. Multi-party ($K > 2$) collaborative learning also involves honest participants who are neither the adversary, nor the target. In the multi-party case, the identities of the participants may not be known to the adversary. Even if the identities are known but the models are aggregated, the adversary may infer something about the training data but not trace it to a specific participant; we discuss this further in Section 9.4.

The updates that adversary observes and uses for inference depend on both K and how collaborative training is done.

As inputs to his inference algorithms, the adversary uses the model updates revealed in each round of collaborative training. For synchronized SGD [52] with $K = 2$, the adversary observes gradient updates computed on a single batch of the target's data. If $K > 2$, he observes an aggregation of gradient updates from all other participants (each computed on a single batch of the respective participant's data). For federated learning with model averaging [35], the observed updates are the result of two-step aggregation: (1) every participant aggregates the gradients computed on each local batch, and (2) the server aggregates the updates from all participants.

For property inference, the adversary needs auxiliary training data correctly labeled with the property he wants to infer (e.g., faces labeled with ages if the goal is to infer ages). For active property inference (Section 4.5), these auxiliary data points must also be labeled for the main task (e.g., faces labeled with identities for a facial recognition model).

4.2 Overview of the attacks

Figure 2 provides a high-level overview of our inference attacks. At each iteration t of training, the adversary downloads the current joint model, calculates gradient updates as prescribed by the collaborative learning algorithm, and sends

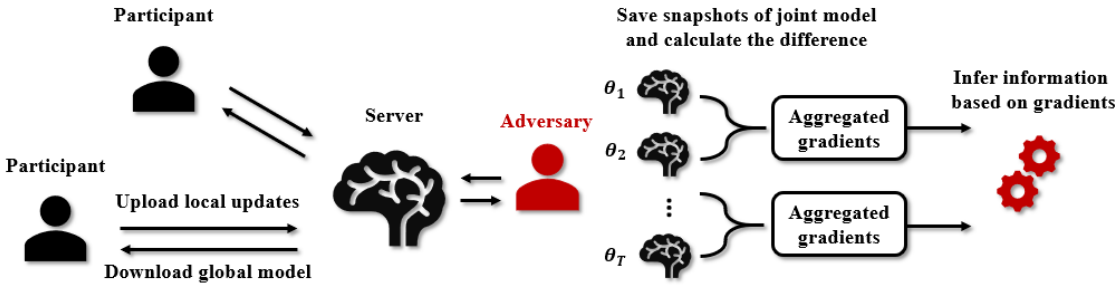


Figure 2: Overview of inference attacks against collaborative learning.

his own updates to the server. The adversary saves the snapshot of the joint model parameters θ_t. The difference between the consecutive snapshots $\Delta\theta_t = \theta_t - \theta_{t-1} = \sum_k \Delta\theta_t^k$ is equal to the aggregated updates from all participants, hence $\Delta\theta_t - \Delta\theta_t^{\text{adv}}$ are the aggregated updates from all participants other than the adversary.

Leakage from the embedding layer. All deep learning models operating on non-numeric data where the input space is discrete and sparse (e.g., natural-language text or locations) first use an embedding layer to transform inputs into a lower-dimensional vector representation. For convenience, we use *word* to denote discrete tokens, i.e., actual words or specific locations. Let vocabulary V be the set of all words. Each word in the training data is mapped to a word-embedding vector via an embedding matrix $W_{\text{emb}} \in \mathbb{R}^{|V| \times d}$, where $|V|$ is the size of the vocabulary and d is the dimensionality of the embedding.

During training, the embedding matrix is treated as a parameter of the model and optimized collaboratively. The gradient of the embedding layer is sparse with respect to the input words: given a batch of text, the embedding is updated only with the words that appear in the batch. The gradients of the other words are zeros. This difference directly reveals which words occur in the training batches used by the honest participants during collaborative learning.

Leakage from the gradients. In deep learning models, gradients are computed by back-propagating the loss through the entire network from the last to the first layer. Gradients of a given layer are computed using this layer's features and the error from the layer above. In the case of sequential fully connected layers h_l, h_{l+1} ($h_{l+1} = W_l \cdot h_l$, where W_l is the weight matrix), the gradient of error E with respect to W_l is computed as $\frac{\partial E}{\partial W_l} = \frac{\partial E}{\partial h_{l+1}} \cdot h_l$. The gradients of W_l are inner products of the error from the layer above and the features h_l. Similarly, for a convolutional layer, the gradients of the weights are convolutions of the error from the layer above and the features h_l. Observations of gradient updates can thus be used to infer feature values, which are in turn based on the participants' private training data.

4.3 Membership inference

As explained above, the non-zero gradients of the embedding layer reveal which words appear in a batch. This helps infer whether a given text or location appears in the training dataset or not. Let V_t be the words included in the updates $\Delta\theta_t$. During training, the attacker collects a vocabulary sequence $[V_1, \ldots, V_T]$. Given a text record r, with words V_r, he can test if $V_r \subseteq V_t$, for some t in the vocabulary sequence. If r is in target's dataset, then V_r will be included in at least one vocabulary from the sequence. The adversary can use this to decide whether r was a member or not.

4.4 Passive property inference

We assume that the adversary has auxiliary data consisting of the data points that have the property of interest ($D^{\text{adv}}_{\text{prop}}$) and data points that do not have the property ($D^{\text{adv}}_{\text{nonprop}}$). These data points need to be sampled from the same class as the target participant's data, but otherwise can be unrelated.

The intuition behind our attack is that the adversary can leverage the snapshots of the global model to generate aggregated updates based on the data with the property and updates based on the data without the property. This produces labeled examples, which enable the adversary to train a binary *batch property classifier* that determines if the observed updates are based on the data with or without the property. This attack is *passive*, i.e., the adversary observes the updates and performs inference without changing anything in the local or global collaborative training procedure.

Batch property classifier. Algorithm 3 shows how to build a batch property classifier during collaborative training. Given a model snapshot θ_t, calculate gradients g_{prop} based on a batch with the property $b^{\text{adv}}_{\text{prop}} \subset D^{\text{adv}}_{\text{prop}}$ and g_{nonprop} based on a batch without the property $b^{\text{adv}}_{\text{nonprop}} \subset D^{\text{adv}}_{\text{nonprop}}$. Once enough labeled gradients have been collected, train a binary classifier f_{prop}.

For the property inference attacks that exploit the embedding-layer gradients (e.g., the attack on the Yelp dataset in Section 6.2), we use a logistic regression classifier. For all other property inference attacks, we experimented with logistic regression, gradient boosting, and random forests. Random forests with 50 trees performed the best. The input features in this case correspond to the observed gradient updates. The number of the features is thus equal to the model's parameters, which can be very large for a realistic model. To downsample the features representation, we apply the max pooling operator [21] on the observed gradient updates. More specifically, max pooling performs a max filter to non-overlapping subregions of the initial features representation, thus reducing the computational cost of the attack.

Inference algorithm. As collaborative training progresses, the

Algorithm 3 Batch Property Classifier

Inputs: Attacker's auxiliary data $D^{\text{adv}}_{\text{prop}}, D^{\text{adv}}_{\text{nonprop}}$
Outputs: Batch property classifier f_{prop}
$G_{\text{prop}} \leftarrow \emptyset$ ▷ Positive training data for property inference
$G_{\text{nonprop}} \leftarrow \emptyset$ ▷ Negative training data for property inference
for $i = 1$ to T **do**
  Receive θ_t from server
  Run **ClientUpdate**(θ_t)
  Sample $b^{\text{adv}}_{\text{prop}} \subset D^{\text{adv}}_{\text{prop}}, b^{\text{adv}}_{\text{nonprop}} \subset D^{\text{adv}}_{\text{nonprop}}$
  Calculate $g_{\text{prop}} = \nabla L(b^{\text{adv}}_{\text{prop}}; \theta_t), g_{\text{nonprop}} = \nabla L(b^{\text{adv}}_{\text{nonprop}}; \theta_t)$
  $G_{\text{prop}} \leftarrow G_{\text{prop}} \cup \{g_{\text{prop}}\}$
  $G_{\text{nonprop}} \leftarrow G_{\text{nonprop}} \cup \{g_{\text{nonprop}}\}$
end for
Label G_{prop} as positive and G_{nonprop} as negative
Train a binary classifier f_{prop} given $G_{\text{prop}}, G_{\text{nonprop}}$

adversary observes gradient updates $g_{\text{obs}} = \Delta\theta_t - \Delta\theta^{\text{adv}}_t$. For single-batch inference, the adversary simply feeds the observed gradient updates to the batch property classifier f_{prop}.

This attack can be extended from single-batch properties to the target's entire training dataset. The batch property classifier f_{prop} outputs a score in [0,1], indicating the probability that a batch has the property. The adversary can use the average score across all iterations to decide whether the target's entire dataset has the property in question.

4.5 Active property inference

An active adversary can perform a more powerful attack by using *multi-task learning*. The adversary extends his local copy of the collaboratively trained model with an augmented property classifier connected to the last layer. He trains this model to simultaneously perform well on the main task and recognize batch properties. On the training data where each record has a main label y and a property label p, the model's joint loss is calculated as:

$$L_{\text{mt}} = \alpha \cdot L(x, y; \theta) + (1 - \alpha) \cdot L(x, p; \theta)$$

During collaborative training, the adversary uploads the updates $\nabla_\theta L_{\text{mt}}$ based on this joint loss, causing the joint model to learn separable representations for the data with and without the property. As a result, the gradients will be separable too (e.g., see Figure 7 in Section 6.5), enabling the adversary to tell if the training data has the property.

This adversary is still "honest-but-curious" in the cryptographic parlance. He faithfully follows the collaborative learning protocol and does not submit any malformed messages. The only difference with the passive attack is that this adversary performs additional *local* computations and submits the resulting values into the collaborative learning protocol. Note that the "honest-but-curious" model does not constrain the parties' input values, only their messages.

5 Datasets and model architectures

The datasets, collaborative learning tasks, and adversarial inference tasks used in our experiments are reported in Table 1.

Dataset	#Records	Main tasks	Inference tasks
LFW	13.2k	Gender/Smile/Age Eyewear/Race/Hair	Race/Eyewear
FaceScrub	18.8k	Gender	Identity
PIPA	18.0k	Age	Gender
CSI	1.4k	Sentiment	Membership, Region/Gender/Veracity
FourSquare	15.5k	Gender	Membership
Yelp-health	17.9k	Review score	Membership, Doctor specialty
Yelp-author	16.2K	Review score	Author

Table 1: Datasets and tasks used in our experiments.

Our choices of hyperparameters are based on the standard models from the ML literature.

Labeled Faces In the Wild (LFW). LFW [28] contains 13,233 62x47 RGB face images for 5,749 individuals with labels such as gender, race, age, hair color, and eyewear.

FaceScrub. FaceScrub [40] contains 76,541 50x50 RGB images for 530 individuals with the gender label: 52.5% are labeled as male, the rest as female. For our experiments, we selected a subset of 100 individuals with the most images, for a total of 18,809 images.

On both LFW and FaceScrub, the collaborative models are convolutional neural networks (CNN) with three spatial convolution layers with 32, 64, and 128 filters, kernel size set to $(3, 3)$, and max pooling layers with pooling size set to 2, followed by two fully connected layers of size 256 and 2. We use rectified linear unit (ReLU) as the activation function for all layers. Batch size is 32 (except in the experiments where we vary it), SGD learning rate is 0.01.

People in Photo Album (PIPA). PIPA [67] contains over 60,000 photos of 2,000 individuals collected from public Flickr photo albums. Each image includes one or more people and is labeled with the number of people and their gender, age, and race. For our experiments, we selected a subset of 18,000 images with three or fewer people and scaled the raw images to 128x128.

The collaborative model for PIPA is a VGG-style [54] 10-layer CNN with two convolution blocks consisting of one convolutional layer and max pooling, followed by three convolution blocks consisting of two convolutional layers and max pooling, followed by two fully connected layers. Batch size is 32, SGD learning rate is 0.01.

Yelp-health. We extracted health care-related reviews from the Yelp dataset[1] of 5 million reviews of businesses tagged with numeric ratings (1-5) and attributes such as business type and location. Our subset contains 17,938 reviews for 10 types of medical specialists.

Yelp-author. We also extracted a Yelp subset with the reviews of the top 10 most prolific reviewers, 16,207 in total.

On both Yelp datasets, the model is a recurrent neural network with a word-embedding layer of dimension 100. Words in a review are mapped to a sequence of word-embedding vectors, which is fed to a gated recurrent unit (GRU [10]) layer

[1] https://www.yelp.com/dataset

that maps it to a sequence of hidden vectors. We add a fully connected classification layer to the last hidden vector of the sequence. SGD learning rate is 0.05.

FourSquare. In [63, 64], Yang et al. collected a global dataset of FourSquare location "check-ins" (userID, time, location, activity) from April 2012 to September 2013. For our experiments, we selected a subset of 15,548 users who checked in at least 10 different locations in New York City and for whom we know their gender [65]. This yields 528,878 check-ins. The model is a gender classifier, a task previously studied by Pang et al. [44] on similar datasets.

CLiPS Stylometry Investigation (CSI) Corpus. This annually expanded dataset [60] contains student-written essays and reviews. We obtained 1,412 reviews, equally split between Truthful/Deceptive or Positive/Negative and labeled with attributes of the author (gender, age, sexual orientation, region of origin, personality profile) and the document (timestamp, genre, topic, veracity, sentiment). 80% of the reviews are written by females, 66% by authors from Antwerpen, the rest from other parts of Belgium and the Netherlands.

On both the FourSquare and CSI datasets, the model, which is based on [31], first uses an embedding layer to turn non-negative integers (locations indices and word tokens) into dense vectors of dimension 320, then applies three spatial convolutional layers with 100 filters and variable kernel windows of size $(3, 320)$, $(4, 320)$ and $(5, 320)$ and max pooling layers with pooling size set to $(l-3, 1)$, $(l-4, 1)$, and $(l-5, 1)$ where l is the fixed length to which input sequences are padded. The hyperparameter l is 300 on CSI and 100 on FourSquare. After this, the model has two fully connected layers of size 128 and 2 for FourSquare and one fully connected layer of size 2 for CSI. We use RELU as the activation function. Batch size is 100 for FourSquare, 12 for CSI. SGD learning rate is 0.01.

6 Two-Party Experiments

All experiments were performed on a workstation running Ubuntu Server 16.04 LTS equipped with a 3.4GHz CPU i7-6800K, 32GB RAM, and an NVIDIA TitanX GPU card. We use MxNet [8] and Lasagne [12] to implement deep neural networks and Scikit-learn [48] for conventional ML models. The source code is available upon request. Training our inference models takes less than 60 seconds on average and does not require a GPU.

We use AUC scores to evaluate the performance of both the collaborative model and our property inference attacks. For membership inference, we report only precision because our decision rule from Section 4.3 is binary and does not produce a probability score.

6.1 Membership inference

The adversary first builds a Bag of Words (BoW) representation for the input whose membership in the target's training data he aims to infer. We denote this as the test BoW. During training, as explained in Section 4.3, the non-zero gradients of the embedding layer reveal which "words" are present in each batch of the target's data, enabling the adversary to build a batch BoW. If the test BoW is a subset of the batch BoW, the adversary infers that the input of interest occurs in the batch.

We evaluate membership inference on the Yelp-health and FourSquare datasets with the vocabulary of 5,000 most frequent words and 30,000 most popular locations, respectively. We split the data evenly between the target and the adversary and train a collaborative model for 3,000 iterations.

Table 2 shows the precision of membership inference for different batch sizes. As batch size increases, the adversary observes more words in each batch BoW and the attack produces more false positives. Recall is always perfect (i.e., no false negatives) because any true test BoW must be contained in at least one of the batch BoWs observed by the adversary.

Yelp-health		FourSquare	
Batch Size	**Precision**	**Batch Size**	**Precision**
32	0.92	100	0.99
64	0.84	200	0.98
128	0.75	500	0.91
256	0.66	1,000	0.76
512	0.62	2,000	0.62

Table 2: Precision of membership inference (recall is 1).

Main T.	Infer T.	Corr.	AUC	Main T.	Infer T.	Corr.	AUC
Gender	Black	-0.005	1.0	Gender	Sunglasses	-0.025	1.0
Gender	Asian	-0.018	0.93	Gender	Eyeglasses	0.157	0.94
Smile	Black	0.062	1.0	Smile	Sunglasses	-0.016	1.0
Smile	Asian	0.047	0.93	Smile	Eyeglasses	-0.083	0.97
Age	Black	-0.084	1.0	Race	Sunglasses	0.026	1.0
Age	Asian	-0.078	0.97	Race	Eyeglasses	-0.116	0.96
Eyewear	Black	0.034	1.0	Hair	Sunglasses	-0.013	1.0
Eyewear	Asian	-0.119	0.91	Hair	Eyeglasses	0.139	0.96

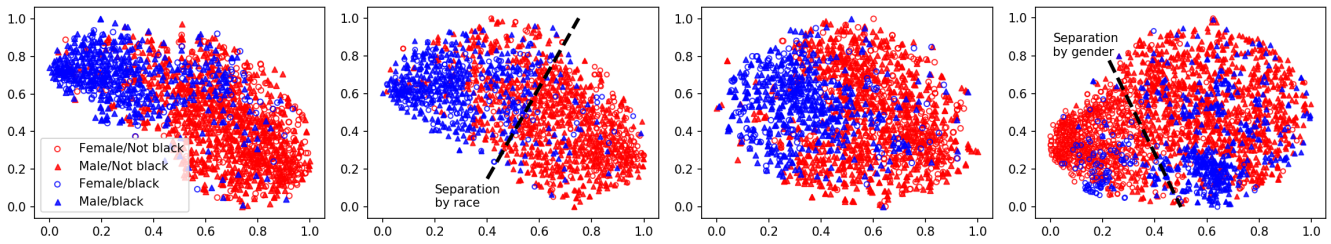
Table 3: AUC score of single-batch property inference on LFW. We also report the Pearson correlation between the main task label and the property label.

6.2 Single-batch property inference

We call a training batch b_{nonprop} if none of the inputs in it have the property, b_{prop} otherwise. The adversary aims to identify which of the batches are b_{prop}. We split the training data evenly between the target and the adversary and assume that the same fraction of inputs in both subsets have the property. During training, $\frac{1}{m}$ of the target's batches include only inputs with the property ($m = 2$ in the following).

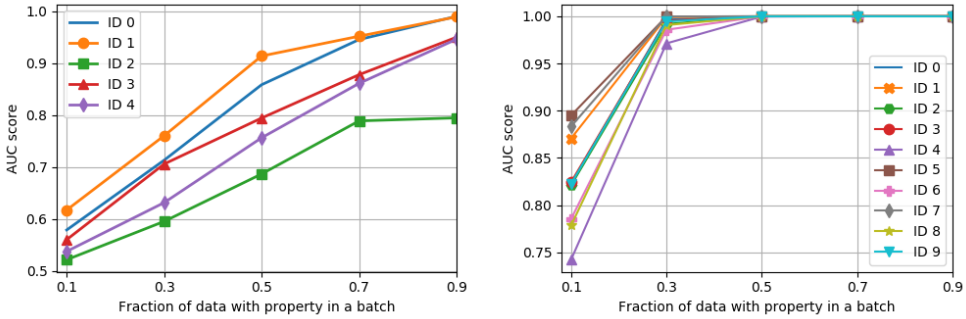
LFW. Table 3 reports the results of single-batch property inference on the LFW dataset. We chose properties that are *uncorrelated* with the main classification label that the collaborative model is trying to learn. The attack has perfect AUC when the main task is gender classification and the inference task is "race:black" (the Pearson correlation between these labels is -0.005). The attack also achieves almost perfect AUC when the main task is "race: black" and the inference task is "eyewear: sunglasses." It also performs well on several other properties, including "eyewear: glasses" when the main task is "race: Asian."

These results demonstrate that gradients observed during training leak more than the characteristic features of each class.



(a) pool1 **(b)** pool2 **(c)** pool3 **(d)** fc

Figure 3: t-SNE projection of the features from different layers of the joint model on LFW gender classification; hollow circle point is female, solid triangle point is male, blue point is the property "race: black" and red point is data without the property.



(a) FaceScrub **(b)** Yelp-author

Figure 4: AUC vs. the fraction of the batch that has the property on FaceScrub and Yelp-author.

In fact, **collaborative learning leaks properties of the training data that are uncorrelated with class membership**. To understand why, we plot the t-SNE projection [59] of the features from different layers of the joint model in Figure 3. Observe that the feature vectors are grouped by property in the lower layers pool1, pool2 and pool3, and by class label in the higher layer. Intuitively, the model did not just learn to separate inputs by class. The lower layers of the model also learned to separate inputs by various properties that are uncorrelated with the model's designated task. Our inference attack exploits this unintended extra functionality.

Yelp-health. On this dataset, we use review-score classification as the main task and the specialty of the doctor being reviewed as the property inference task. Obviously, the latter is more sensitive from the privacy perspective.

We use 3,000 most frequent words in the corpus as the vocabulary and train for 3,000 iterations. Using BoWs from the embedding-layer gradients, the attack achieves almost perfect AUC. Table 4 shows the words that have the highest predictive power in our logistic regression.

Fractional properties. We now attempt to infer that *some* of the inputs in a batch have the property. For these experiments, we use FaceScrub's top 5 face IDs and Yelp-author (the latter with the 3,000 most frequent words as the vocabulary). The model is trained for 3,000 iterations. As before, $1/2$ of the target's batches include inputs with the property, but here we vary the fraction of inputs with the property within each such batch among 0.1, 0.3, 0.5, 0.7, and 0.9.

Figure 4 reports the results. On FaceScrub for IDs 0, 1, and 3, AUC scores are above 0.8 even if only 50% of the batch contain that face, i.e., the adversary can successfully infer that photos of a particular person appear in a batch even though (a) the model is trained for generic gender classification, and (b) half of the photos in the batch are of other people. If the fraction is higher, AUC approaches 1.

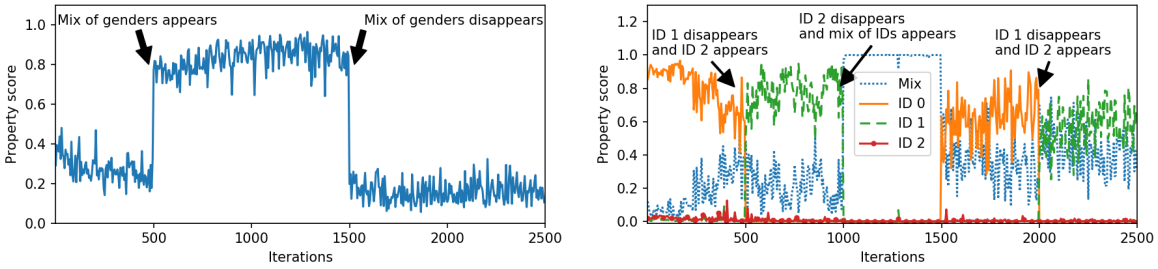
Health Service	Top Words in Positive Class
Obstetricians	pregnancy, delivery, women, birth, ultrasound
Pediatricians	pediatrics, sick, parents, kid, newborn
Cosmetic Surgeons	augmentation, plastic, breast, facial, implants
Cardiologists	cardiologist, monitor, bed, heart, ER
Dermatologists	acne, dermatologists, mole, cancer, spots
Ophthalmologists	vision, LASIK, contacts, lenses, frames
Orthopedists	knee, orthopedic, shoulder, injury, therapy
Radiologists	imaging, SimonMed, mammogram, CT, MRI
Psychiatrists	psychiatrist, mental, Zedek, depression, sessions
Urologists	Edgepark, pump, supplies, urologist, kidney

Table 4: Words with the largest positive coefficients in the property classifier for Yelp-health.

On Yelp-author, AUC scores are above 0.95 for all identities even when the fraction is 0.3, i.e., the attack successfully infers the authors of reviews even though (a) the model is trained for generic sentiment analysis, and (b) more than two thirds of the reviews in the batch are from other authors.

6.3 Inferring when a property occurs

Continuous training, when new training data is added to the process as it becomes available, presents interesting opportunities for inference attacks. If the occurrences of a property in the training data can be linked to events outside the training process, privacy violation is exacerbated. For example, sup-



(a) PIPA **(b)** FaceScrub

Figure 5: Inferring occurrence of a single-batch property.

pose a model leaks that a certain third person started appearing in another participant's training data immediately after that participant uploaded his photos from a trip.

PIPA. Images in the PIPA dataset have between 1 to 3 faces. We train the collaborative model to detect if there is a young adult in the image; the adversary's inference task is to determine if people in the image are of the same gender. The latter property is a stepping stone to inferring social relationships and thus sensitive. We train the model for 2,500 iterations and let the batches with the "same gender" property appear in iterations 500 to 1500.

Figure 5a shows, for each iteration, the probability output by the adversary's classifier that the batch in that iteration has the property. The appearance and disappearance of the property in the training data are clearly visible in the plot.

FaceScrub. For the gender classification model on FaceScrub, the adversary's objective is to infer whether and when a certain person appears in the other participant's photos. The joint model is trained for 2,500 iterations. We arrange the target's training data so that two specific identities appear during certain iterations: ID 0 in iterations 0 to 500 and 1500 to 2000, ID 1 in iterations 500 to 1000 and 2000 to 2500. The rest of the batches are mixtures of other identities. The adversary trains three property classifiers, for ID 0, ID 1, and also for ID 2 which does not appear in the target's dataset.

Figure 5b reports the scores of all three classifiers. ID 0 and 1 receive the highest scores in the iterations where they appear, whereas ID 2, which never appears in the training data, receives very low scores in all iterations.

These experiments show that our attacks can successfully infer *dynamic properties* of the training dataset as collaborative learning progresses.

6.4 Inference against well-generalized models

To show that our attacks work with (1) relatively few observed model updates and (2) against well-generalized models, we experiment with the CSI corpus. Figure 6 reports the accuracy of inferring the author's gender. The attack reaches 0.98 AUC after only 2 epochs and improves as the training progresses and the adversary collects more updates.

Figure 6 also shows that the model is not overfitted. Its test accuracy on the main sentiment-analysis task is high and improves with the number of the epochs.

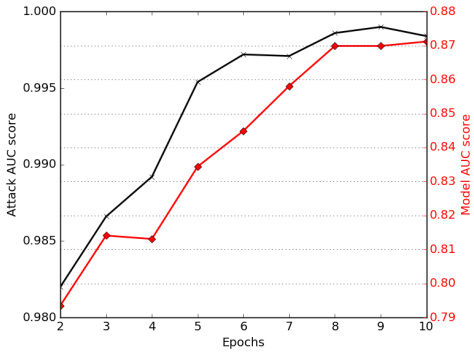


Figure 6: Attack performance with respect to the number of collaborative learning epochs.

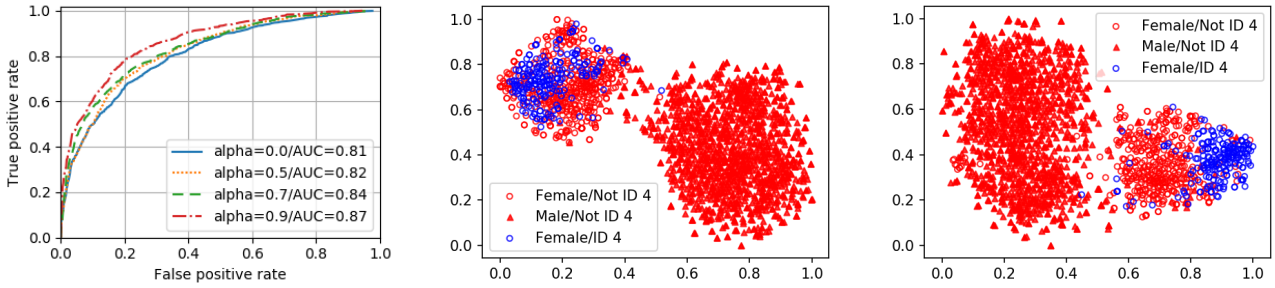
6.5 Active property inference

To show the additional power of the active attack from Section 4.5, we use FaceScrub. The main task is gender classification, the adversary's task is to infer the presence of ID 4 in the training data. We assume that this ID occurs in a single batch, where it constitutes 50% of the photos. We evaluate the attack with different choices of α, which controls the balance between the main-task loss and the property-classification loss in the adversary's objective function.

Figure 7a shows that AUC increases as we increase α. Figure 7b and Figure 7c show the t-SNE projection of the final fully connected layer, with $\alpha = 0$ and $\alpha = 0.7$, respectively. Observe that the data with the property (blue points) is grouped tighter when $\alpha = 0.7$ than in the model trained under a passive attack ($\alpha = 0$). This illustrates that *as a result of the active attack, the joint model learns a better separation for data with and without the property*.

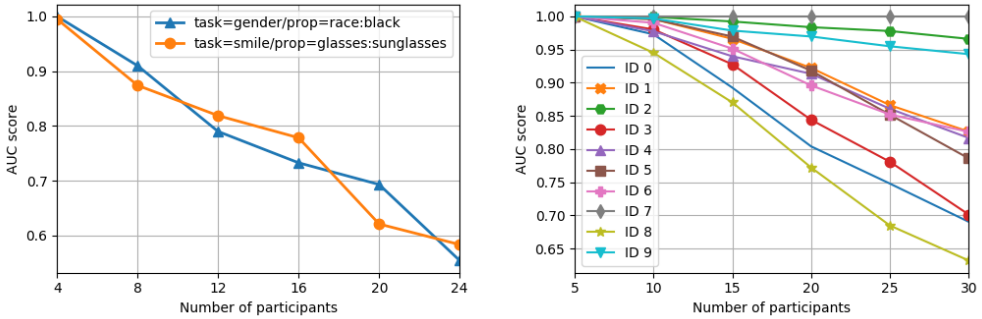
7 Multi-Party Experiments

In the multi-party setting, we only consider passive property inference attacks. We vary the number of participants between 4 and 30 to match the deployment scenarios and applications proposed for collaborative learning, e.g., hospitals or biomedical research institutions training on private medical data [29, 30]. This is similar to prior work [25], which was evaluated on MNIST with 2 participants and face recognition on the AT&T dataset with 41 participants.



(a) ROC for different α **(b)** t-SNE of the final layer for $\alpha = 0$ **(c)** t-SNE of the final layer for $\alpha = 0.7$.

Figure 7: Two-party active property inference attack on FaceScrub. For (b) and (c), hollow circle point is female, solid triangle point is male, blue point is the property "ID 4" and red point is data without the property.



(a) LFW **(b)** Yelp-author

Figure 8: Multi-party learning with synchronized SGD: attack AUC score vs. the number of participants.

7.1 Synchronized SGD

As the number of honest participants in collaborative learning increases, the adversary's task becomes harder because the observed gradient updates are aggregated across multiple participants. Furthermore, the inferred information may not directly reveal the identity of the participant to whom the data belongs (see Section 9.4).

In the following experiments, we split the training data evenly across all participants, but so that only the target and the adversary have the data with the property. The joint model is trained with the same hyperparameters as in the two-party case. Similar to Section 6.2, the adversary's goal is to identify which aggregated gradient updates are based on batches b_{prop} with the property.

LFW. We experiment with (1) gender classification as the main task and "race: black" as the inference task, and (2) smile classification as the main task and "eyewear: sunglasses" as the inference task. Figure 8a shows that the attack still achieves reasonably high performance, with AUC score around 0.8, when the number of participants is 12. Performance then degrades for both tasks.

Yelp-author. The inference task is again author identification. In the multi-party case, the gradients of the embedding layer leak the batch BoWs of all honest participants, not just the target. Figure 8b reports the results. For some authors, AUC scores do not degrade significantly even with many participants. This is likely due to some unique combinations of words used by these authors, which identify them even in multi-party settings.

7.2 Model averaging

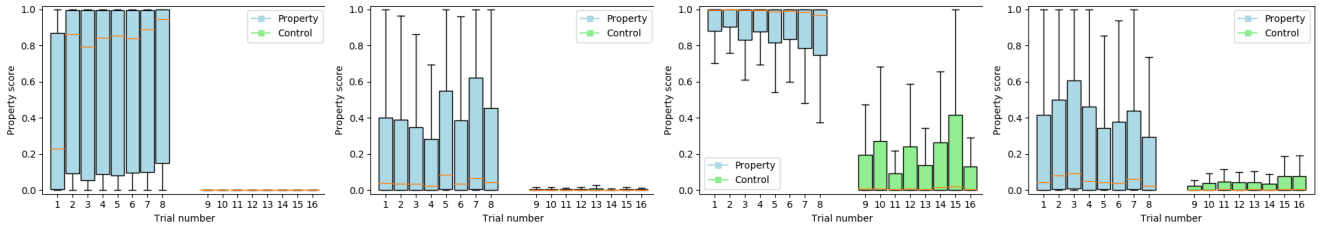
In every round t of federated learning with model averaging (see Algorithm 2), the adversary observes $\theta_t - \theta_{t-1} = \sum_k \frac{n^k}{n}\theta_t^k - \sum_k \frac{n^k}{n}\theta_{t-1}^k = \sum_k \frac{n^k}{n}(\theta_t^k - \theta_{t-1}^k)$, where $\theta_t^k - \theta_{t-1}^k$ are the aggregated gradients computed on the k-th participant's local dataset.

In our experiments, we split the training data evenly among honest participants but ensure that in the target participant's subset, $\hat{p}\%$ of the inputs have the property, while none of the other honest participants' data have it. During each epoch of local training, every honest participant splits his local training data into 10 batches and performs one round of training.

We assume that the adversary has the same number of inputs with the property as the target. As before, when the adversary trains his binary classifier, he needs to locally "emulate" the collaborative training process, i.e., sample data from his local dataset, compute aggregated updates, and learn to distinguish between the aggregates based on the data without the property and aggregates where one of the underlying updates was based on the data with the property.

We perform 8 trials where a subset of the training data has the property and 8 control trials where there are no training inputs with the property.

Inferring presence of a face. We use FaceScrub and select two face IDs (1 and 3) whose presence we want to infer. In the "property" case, $\hat{p} = 80\%$, i.e., 80% of one honest participant's training data consist of the photos that depict the person in question. In the control case, $\hat{p} = 0\%$, i.e., the photos of this person do not occur in the training data. Figure 9 shows



(a) Face ID 1, $K = 3$ **(b)** Face ID 1, $K = 5$ **(c)** Face ID 3, $K = 3$ **(d)** Face ID 3, $K = 5$

Figure 9: Multi-party learning with model averaging. Box plots show the distribution of the adversary's scores in each trial: in the 8 trials on the left, one participant's data has the property; in the 8 trials on the right, none of the honest participants have the data with the property.

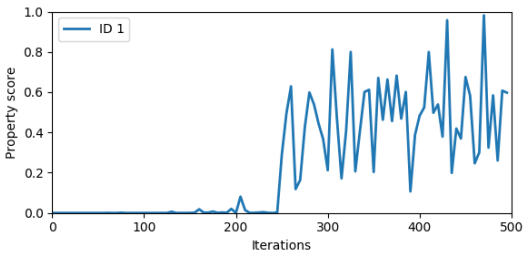


Figure 10: Inferring that a participant whose local data has the property of interest has joined the training. $K = 2$ for rounds 0 to 250, $K = 3$ for rounds 250 to 500.

the scores assigned by the adversary's classifier to the aggregated updates with 3 and 5 total participants. When the face in question is present in the training dataset, the scores are much higher than when it is absent.

Success of the attack depends on the property being inferred, distribution of the data across participants, and other factors. For example, the classifiers for Face IDs 2 and 4, which were trained in the same fashion as the classifiers for Face IDs 1 and 3, failed to infer the presence of the corresponding faces in the training data.

Inferring when a face occurs. In this experiment, we aim to infer when a participant whose local data has a certain property joined collaborative training. We first let the adversary and the rest of the honest participants train the joint model for 250 rounds. The target participant then joins the training at round $t = 250$ with the local data that consists of photos depicting ID 1. Figure 10 reports the results of the experiment: the adversary's AUC scores are around 0 when face ID 1 is not present and then increase almost to 1.0 right after the target participant joins the training.

8 Defenses

8.1 Sharing fewer gradients

As suggested in [52], participants in collaborative learning could share only a fraction of their gradients during each update. This reduces communication overhead and, potentially, leakage, since the adversary observes fewer gradients.

To evaluate this defense, we measure the performance of single-batch inference against a sentiment classifier collaboratively trained on the CSI Corpus by two parties who exchange only a fraction of their gradients. Table 5 shows the resulting AUC scores: when inferring the region of the texts' authors, our attack still achieves 0.84 AUC when only 10% of the updates are shared during each iteration, compared to 0.93 AUC when all updates are shared.

Property / % parameters update	**10%**	**50%**	**100%**
Top region (Antwerpen)	0.84	0.86	0.93
Gender	0.90	0.91	0.93
Veracity	0.94	0.99	0.99

Table 5: Inference attacks against the CSI Corpus for different fractions of gradients shared during training.

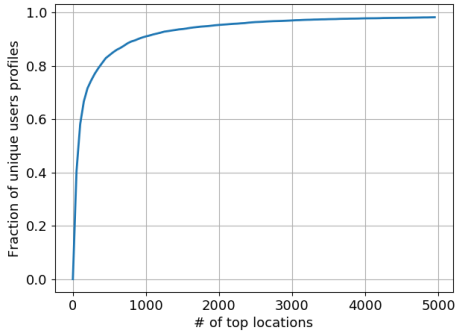


Figure 11: Uniqueness of user profiles with respect to the number of top locations.

8.2 Dimensionality reduction

As discussed in Section 4.2, if the input space of the model is sparse and inputs must be embedded into a lower-dimensional space, non-zero gradient updates in the embedding layer reveal which inputs are present in the training batch.

One plausible defense is to only use inputs that occur many times in the training data. This does not work in general, e.g., Figure 11 shows that restricting inputs to the top locations in the FourSquare dataset eliminates most of the training data.

A smarter defense is to restrict the model so that it only uses "words" from a pre-defined vocabulary of common words. For example, Google's federated learning for predictive keyboards uses a fixed vocabulary of 5,000 words [35]. In Table 6, we report the accuracy of our membership inference attack and the accuracy of the joint model on its main task—gender classification for the FourSquare dataset, sentiment analysis for the CSI Corpus—for different sizes of the common vocabulary (locations and words, respectively). This approach partially mitigates our attacks but also has a significant negative impact on the quality of the collaboratively trained models.

CSI			FourSquare		
Top N words	Attack Precision	Model AUC	Top N locations	Attack Precision	Model AUC
4,000	0.94	0.91	30,000	0.91	0.64
2,000	0.92	0.87	10,000	0.86	0.59
1,000	0.92	0.85	3,000	0.65	0.51
500	0.82	0.84	1,000	0.52	0.50

Table 6: Membership inference against the CSI Corpus and FourSquare for different vocabulary sizes.

Dropout Prob.	Attack AUC	Model AUC
0.1	0.94	0.87
0.3	0.97	0.87
0.5	0.98	0.87
0.7	0.99	0.86
0.9	0.99	0.84

Table 7: Inference of the top region (Antwerpen) on the CSI Corpus for different values of dropout probability.

8.3 Dropout

Another possible defense is to employ *dropout* [56], a popular regularization technique used to mitigate overfitting in neural networks. Dropout randomly deactivates activations between neurons, with probability $p_{drop} \in [0, 1]$. Random deactivations may weaken our attacks because the adversary observes fewer gradients corresponding to the active neurons.

To evaluate this approach, we add dropout after the max pool layers in the joint model. Table 7 reports the accuracy of inferring the region of the reviews in the CSI Corpus, for different values of p_{drop}. Increasing the randomness of dropout makes our attacks *stronger* while slightly decreasing the accuracy of the joint model. Dropout stochastically removes features at every collaborative training step, thus yielding more *informative* features (similar to feature bagging [7, 26]) and increasing variance between participants' updates.

8.4 Participant-level differential privacy

As discussed in Section 2.2, record-level ε-differential privacy, by definition, bounds the success of membership inference but does not prevent property inference. Any application of differential privacy entails application-specific tradeoffs between privacy of the training data and accuracy of the resulting model. The participants must also somehow choose the parameters (e.g., ε) that control this tradeoff.

In theory, participant-level differential privacy bounds the success of inference attacks described in this paper. We implemented the participant-level differentially private federated learning algorithm by McMahan et al. [36] and attempted to train a gender classifier on LFW, but the model did not converge for any number of participants (we tried at most 30). This is due to the magnitude of noise needed to achieve differential privacy with the moments accountant bound [1], which is inversely proportional to the number of users (the model in [36] was trained on *thousands* of users). Another participant-level differential privacy mechanism, presented in [20], also requires a very large number of participants. Moreover, these two mechanisms have been used, respectively, for language modeling [36] and handwritten digit recognition [20]. Adapting them to the specific models and tasks considered in this paper may not be straightforward.

Following [20, 36], we believe that participant-level differential privacy provide reasonable accuracy only in settings involving at least thousands of participants. We believe that further work is needed to investigate whether participant-level differential privacy can be adapted to prevent our inference attacks *and* obtain high-quality models in settings that do not involve thousands of users.

9 Limitations of the attacks

9.1 Auxiliary data

Our property inference attacks assume that the adversary has auxiliary training data correctly labeled with the property he wants to infer. For generic properties, such data is easy to find. For example, the auxiliary data for inferring the number and genders of people can be any large dataset of images with males and females, single and in groups, where each image is labeled with the number of people in it and their genders. Similarly, the auxiliary data for inferring the medical specialty of doctors can consist of any texts that include words characteristic of different specialties (see Table 4).

More targeted inference attacks require specialized auxiliary data that may not be available to the adversary. For example, to infer that photos of a certain person occurs in another participant's dataset, the adversary needs (possibly different) photos of that person to train on. To infer the authorship of training texts, the adversary needs a sufficiently large sample of texts known to be written by a particular author.

9.2 Number of participants

In our experiments, the number of participants in collaborative training is relatively small (ranging from 2 to 30), while some federated-learning applications involve thousands or millions of users [35, 36]. As discussed in Section 7.1, performance of our attacks drops significantly as the number of participants increases.

9.3 Undetectable properties

It may not be possible to infer some properties from model updates. For example, our attack did not detect the presence of some face identities in the multi-party model averaging experiments (Section 7.2). If for whatever reason the model does not internally separate the features associated with the target property, inference will fail.

9.4 Attribution of inferred properties

In the two-party scenarios considered in Section 6, attribution of the inferred properties is trivial because there is only one honest participant. In the multi-party scenarios considered in Section 7, model updates are aggregated. Therefore, even if the adversary successfully infers the presence of inputs with a certain property in the training data, he may not be able to attribute these inputs to a specific participant. Furthermore, he may not be able to tell if all inputs with the property belong to one participant or are distributed across multiple participants.

In general, attribution requires auxiliary information specific to the leakage. For example, consider face identification. In some applications of collaborative learning, the identities of all participants are known because they need to communicate with each other. If collaborative learning leaks that a particular person appears in the training images, auxiliary information about the participants (e.g., their social networks) can reveal which of them knows the person in question. Similarly, if collaborative learning leaks the authorship of the training texts, auxiliary information can help infer which participant is likely to train on texts written by this author.

Another example of attribution based on auxiliary information is described in Section 6.3. If photos of a certain person first appear in the training data after a new participant has joined collaborative training, the adversary may attribute these photos to the new participant.

Note that leakage of medical conditions, locations, images of individuals, or texts written by known authors is a privacy breach even if it cannot be traced to a specific participant or multiple participants. Leaking that a certain person appears in the photos or just the number of people in the photos reveals intimate relationships between people. Locations can reveal people's addresses, religion, sexual orientation, and relationships with other people.

10 Related Work

Privacy-preserving distributed learning. Transfer learning in combination with differentially private (DP) techniques tailored for deep learning [1] has been used in [45, 46]. These techniques privately train a "student" model by transferring, through noisy aggregation, the knowledge of an ensemble of "teachers" trained on the disjoint subsets of training data. These are centralized, record-level DP mechanisms with a trusted aggregator and do not apply to federated or collaborative learning. In particular, [45, 46] assume that the adversary cannot see the individual models, only the final model trained by the trusted aggregator. Moreover, record-level DP by definition does not prevent property inference. Finally, their effectiveness has been demonstrated only on a few specific tasks (MNIST, SVHN, OCR), which are substantially different from the tasks considered in this paper.

Shokri and Shmatikov [52] propose making gradient updates differentially private to protect the training data. Their approach requires extremely large values of the ε parameter (and consequently little privacy protection) to produce an accurate joint model. More recently, participant-level differentially private federated learning methods [20, 36] showed how to protect participants' training data by adding Gaussian noise to local updates. As discussed in Section 8.4, these approaches require a large number of users (on the order of thousands) for the training to converge and achieve an acceptable trade-off between privacy and model performance. Furthermore, the results in [36] are reported for a specific language model and use *AccuracyTop1* as the proxy, not the actual accuracy of the non-private model.

Pathak et al. [47] present a differentially private global classifier hosted by a trusted third-party and based on locally trained classifiers held by separate, mutually distrusting parties. Hamm et al. [23] use knowledge transfer to combine a collection of models trained on individual devices into a single model, with differential privacy guarantees.

Secure multi-party computation (MPC) has also been used to build privacy-preserving neural networks in a distributed fashion. For example, SecureML [37] starts with the data owners (clients) distributing their private training inputs among two non-colluding servers during the setup phase; the two servers then use MPC to train a global model on the clients' encrypted joint data. Bonawitz et al. [5] use secure multi-party aggregation techniques, tailored for federated learning, to let participants encrypt their updates so that the central parameter server only recovers the sum of the updates. In Section 7.2, we showed that inference attacks can be successful even if the adversary only observes aggregated updates.

Membership inference. Prior work demonstrated the feasibility of membership inference from aggregate statistics, e.g., in the context of genomic studies [3, 27], location time-series [50], or noisy statistics in general [14].

Membership inference against black-box ML models has also been studied extensively in recent work. Shokri et al. [53] demonstrate membership inference against black-box supervised models, exploiting the differences in the models' outputs on training and non-training inputs. Hayes et al. [24] focus on generative models in machine-learning-as-a-service applications and train GANs [22] to detect overfitting and recognize training inputs. Long et al. [34] and Yeom et al. [66] study the relationship between overfitting and information leakage.

Truex et al. [58] extend [53] to a more general setting and show how membership inference attacks are data-driven and largely transferable. They also show that an adversary who participates in collaborative learning, with access to individual model updates from all honest participants, can boost the performance of membership inference vs. a centralized model. Nasr et al. [39] design a privacy mechanism to adversarially train *centralized* machine learning models with provable protections against membership inference.

Other attacks on machine learning models. Several techniques infer class features and/or construct class representatives if the adversary has black-box [16, 17] or white-box [2] access to a classifier model. As discussed in detail in Section 3, these techniques infer features that characterize an entire class and not specifically the training data, except in the cases of pathological overfitting where the training sample constitutes the entire membership of the class.

Hitaj et al. [25] show that a participant in collaborative deep learning can use GANs to construct class representatives. Their technique was evaluated only on models where all members of the same class are visually similar (handwritten digits and faces). As discussed in Section 3.1, there is no evidence that it produces actual training images or can distinguish a training image and another image from the same class.

The informal property violated by the attacks of [2, 16, 17, 25] is: "a classifier should prevent users from generating an input that belongs to a particular class or even learning what

such an input looks like." It is not clear to us why this property is desirable, or whether it is even achievable.

Aono et al. [49] show that, in the collaborative deep learning protocol of [52], an honest-but-curious server can partially recover participants' training inputs from their gradient updates under the (greatly simplified) assumption that the batch consists of a single input. Furthermore, the technique is evaluated only on MNIST where all class members are visually similar. It is not clear if it can distinguish a training image and another image from the same MNIST class.

Song et al. [55] engineer an ML model that memorizes the training data, which can then be extracted with black-box access to the model. Carlini et al. [6] show that deep learning-based generative sequence models trained on text data can unintentionally memorize training inputs, which can then be extracted with black-box access. They do this for sequences of digits artificially introduced into the text, which are not affected by the relative word frequencies in the language model.

Training data that is explicitly incorporated or otherwise memorized in the model can also be leaked by model stealing attacks [41, 57, 61].

Concurrently with this work, Ganju et al. [18] developed property inference attacks against fully connected, relatively shallow neural networks. They focus on the post-training, white-box release of models trained on sensitive data, as opposed to collaborative training. In contrast to our attacks, the properties inferred in [18] may be correlated with the main task. Evaluation is limited to simple datasets and tasks such as MNIST, U.S. Census tabular data, and hardware performance counters with short features.

11 Conclusion

In this paper, we proposed and evaluated several inference attacks against collaborative learning. These attacks enable a malicious participant to infer not only *membership*, i.e., the presence of exact data points in other participants' training data, but also *properties* that characterize subsets of the training data and are independent of the properties that the joint model aims to capture.

Deep learning models appear to internally recognize many features of the data that are uncorrelated with the tasks they are being trained for. Consequently, model updates during collaborative learning leak information about these "unintended" features to adversarial participants. Active attacks are potentially very powerful in this setting because they enable the adversary to trick the joint model into learning features of the adversary's choosing without a significant impact on the model's performance on its main task.

Our results suggest that leakage of unintended features exposes collaborative learning to powerful inference attacks. We also showed that defenses such as selective gradient sharing, reducing dimensionality, and dropout are not effective. This should motivate future work on better defenses. For instance, techniques that learn only the features relevant to a given task [15, 42, 43] can potentially serve as the basis for "least-privilege" collaboratively trained models. Further, it may be possible to detect active attacks that manipulate the model into learning extra features. Finally, it remains an open question if participant-level differential privacy mechanisms can produce accurate models when collaborative learning involves relatively few participants.

Acknowledgments. This research was supported in part by the NSF grants 1611770 and 1704296, the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program, the Alan Turing Institute under the EPSRC grant EP/N510129/1, and a grant by Nokia Bell Labs.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *CCS*, 2016.

[2] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN*, 10(3):137–150, 2015.

[3] M. Backes, P. Berrang, M. Humbert, and P. Manoharan. Membership Privacy in MicroRNA-based Studies. In *CCS*, 2016.

[4] BBC. Google DeepMind NHS app test broke UK privacy law. https://bbc.in/2A1MftK, 2017.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.

[6] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song. The Secret Sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv:1802.08232*, 2018.

[7] C.-H. Chang, L. Rampasek, and A. Goldenberg. Dropout feature ranking for deep learning models. *arXiv:1712.08645*, 2017.

[8] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv:1512.01274*, 2015.

[9] T. M. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project Adam: Building an efficient and scalable deep learning training system. In *OSDI*, 2014.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[11] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. Le, et al. Large scale distributed deep networks. In *NIPS*, 2012.

[12] S. Dieleman, J. Schlüter, C. Raffel, et al. Lasagne: First release. http://dx.doi.org/10.5281/zenodo.27878, 2015.

[13] C. Dwork and M. Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1):93–107, 2010.

[14] C. Dwork, A. Smith, T. Steinke, J. Ullman, and S. Vadhan. Robust traceability from trace amounts. In *FOCS*, 2015.

[15] H. Edwards and A. Storkey. Censoring representations with an adversary. In *ICLR*, 2016.

[16] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 2015.

[17] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized Warfarin dosing. In *USENIX Security*, 2014.

[18] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS*, 2018.

[19] General Data Protection Regulation. https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, 2018.

[20] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *arXiv:1712.07557*, 2017.

[21] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep Learning*. MIT Press, 2016.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[23] J. Hamm, Y. Cao, and M. Belkin. Learning privately from multiparty data. In *ICML*, 2016.

[24] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro. LOGAN: Membership inference attacks against generative models. In *PETS*, 2019.

[25] B. Hitaj, G. Ateniese, and F. Pérez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. In *CCS*, 2017.

[26] T. K. Ho. Random decision forests. In *DAR*, 1995.

[27] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8), 2008.

[28] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07–49, University of Massachusetts, Amherst, 2007.

[29] A. Jochems, T. M. Deist, I. El Naqa, M. Kessler, C. Mayo, J. Reeves, S. Jolly, M. Matuszak, R. Ten Haken, J. van Soest, et al. Developing and validating a survival prediction model for NSCLC patients through distributed learning across 3 countries. *Int J Radiat Oncol Biol Phys*, 99(2):344–352, 2017.

[30] A. Jochems, T. M. Deist, J. Van Soest, M. Eble, P. Bulens, P. Coucke, W. Dries, P. Lambin, and A. Dekker. Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital–a real life proof of concept. *Radiother Oncol*, 121(3):459–467, 2016.

[31] Y. Kim. Convolutional neural networks for sentence classification. *arXiv:1408.5882*, 2014.

[32] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 2015.

[33] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*, 2018.

[34] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen. Understanding membership inferences on well-generalized learning models. *arXiv:1802.04889*, 2018.

[35] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[36] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private language models without losing accuracy. In *ICLR*, 2018.

[37] P. Mohassel and Y. Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *S&P*, 2017.

[38] P. Moritz, R. Nishihara, I. Stoica, and M. I. Jordan. SparkNet: Training deep networks in Spark. *arXiv:1511.06051*, 2015.

[39] M. Nasr, R. Shokri, and A. Houmansadr. Machine learning with membership privacy using adversarial regularization. In *CCS*, 2018.

[40] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, 2014.

[41] S. J. Oh, M. Augustin, M. Fritz, and B. Schiele. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[42] S. A. Osia, A. S. Shamsabadi, A. Taheri, K. Katevas, S. Sajadmanesh, H. R. Rabiee, N. D. Lane, and H. Haddadi. A hybrid deep learning architecture for privacy-preserving mobile analytics. *arXiv:1703.02952*, 2017.

[43] S. A. Osia, A. Taheri, A. S. Shamsabadi, K. Katevas, H. Haddadi, and H. R. Rabiee. Deep private-feature extraction. *TKDE*, 2019.

[44] J. Pang and Y. Zhang. DeepCity: A feature learning framework for mining location check-ins. In *ICWSM (Poster Papers)*, 2017.

[45] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.

[46] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with PATE. In *ICLR*, 2018.

[47] M. Pathak, S. Rane, and B. Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *NIPS*, 2010.

[48] 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*, 12, 2011.

[49] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning: Revisited and enhanced. In *ATIS*, 2017.

[50] A. Pyrgelis, C. Troncoso, and E. De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *NDSS*, 2018.

[51] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

[52] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *CCS*, 2015.

[53] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *S&P*, 2017.

[54] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[55] C. Song, T. Ristenpart, and V. Shmatikov. Machine learning models that remember too much. In *CCS*, 2017.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[57] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security*, 2016.

[58] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei. Towards

demystifying membership inference attacks. *arXiv:1807.09173*, 2018.

[59] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 2008.

[60] B. Verhoeven and W. Daelemans. CLiPS Stylometry Investigation (CSI) Corpus: A Dutch corpus for the detection of age, gender, personality, sentiment and deception in text. In *LREC*, 2014.

[61] B. Wang and N. Z. Gong. Stealing hyperparameters in machine learning. In *S&P*, 2018.

[62] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 2015.

[63] D. Yang, D. Zhang, L. Chen, and B. Qu. NationTelescope: Monitoring and visualizing large-scale collective behavior in LBSNs. *JNCA*, 55:170–180, 2015.

[64] D. Yang, D. Zhang, and B. Qu. Participatory cultural mapping based on collective behavior in location based social networks. *ACM TIST*, 2015.

[65] D. Yang, D. Zhang, B. Qu, and P. Cudre-Mauroux. PrivCheck: Privacy-preserving check-in data publishing for personalized location based services. In *UbiComp*, 2016.

[66] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*, 2018.

[67] N. Zhang, M. Paluri, Y. Taigman, R. Fergus, and L. Bourdev. Beyond frontal faces: Improving person recognition using multiple cues. In *CVPR*, 2015.

[68] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *NIPS*, 2010.