InstaHide: Instance-hiding Schemes for Private Distributed Learning

Yangsibo Huang [1] Zhao Song [2] Kai Li [1] Sanjeev Arora [2]

Abstract

How can multiple distributed entities collaboratively train a shared deep net on their private data while preserving privacy? This paper introduces *InstaHide*, a simple encryption of training images, which can be plugged into an existing distributed deep learning pipeline. The encryption is efficient and has minor effect on test accuracy.

InstaHide encrypts each training image with a "one-time secret key" which consists of mixing a number of randomly chosen images and applying a random pixel-wise mask. Other contributions of this paper include: (a) Using a large public dataset (e.g. ImageNet) for mixing during its encryption, which improves security. (b) Experimental results to show effectiveness in preserving privacy against known attacks with only minor effects on accuracy. (c) Theoretical analysis showing that successfully attacking privacy requires attackers to solve a difficult computational problem. (d) Demonstrating that *Mixup* alone is insecure (as contrary to recent proposals), by presenting some efficient attacks. (e) Release of a challenge dataset[1] to encourage new attacks.

1. Introduction

In many applications, multiple parties or clients with sensitive data want to collaboratively train a neural network. For instance, hospitals may wish to train a model on their patient data. However, aggregating data to a central server may violate regulations such as Health Insurance Portability and Accountability Act (HIPAA) (Act, 1996) and General Data Protection Regulation (GDPR) (Voigt & Von dem Bussche, 2018).

Federated learning (McMahan et al., 2016; Konečný et al., 2016) proposes letting participants train on their own data in a distributed fashion and share only model updates — i.e., gradients—with the central server. The server aggregates these updates (typically by averaging) to improve a global model and then sends updates to participants. This process runs iteratively until the global model converges. Merging information from individual data points into aggregated gradients intuitively preserves privacy to some degree. On top of that, it is possible to add noise to gradients in accordance with Differential Privacy (DP) (Dwork et al., 2006; Dwork & Roth, 2014), though careful calculations are needed to compute the amount of noise to be added (Abadi et al., 2016; Papernot et al., 2019). However, the privacy guarantee of DP only applies to the trained model (i.e., approved use of data) and does not apply to side-channel computations performed by curious/malicious parties who are privy to the communicated gradients. Recent work (Zhu et al., 2019) suggests that eavesdropping attackers can recover private inputs from shared model updates, even when DP was used. A more serious issue with DP is that meaningful guarantees involve adding so much noise that test accuracy reduces by over 20% even on CIFAR-10 (Papernot et al., 2019).

Cryptographic methods such as *secure multiparty computation* of (Yao, 1982) and fully-homomorphic encryption (Gentry, 2009) can ensure privacy against arbitrary side-computations by adversary during training. Unfortunately it is a challenge to use them in modern deep learning settings, owing to their high computational overheads and their needs for special setups (e.g finite field arithmetic, public-key infrastructure).

Here we introduce a new method *InstaHide*, inspired by a weaker cryptographic idea of *instance hiding* schemes (Abadi et al., 1987). We only apply it to image data in this paper and leave other data types (e.g., text) for future work. *InstaHide* gives a way to transform input x to a hidden/encrypted input $\widetilde{x}$ in each epoch such that: (a) Training deep nets using the $\widetilde{x}$'s instead of x's gives nets almost as good in terms of final accuracy; (b) Known methods for recovering information about x out of $\widetilde{x}$ are computationally very expensive. In other words, $\widetilde{x}$ effectively hides information contained in x except for its label.

InstaHide encryption has two key components. The first is inspired by *Mixup* data augmentation method (Zhang et al.,

[1]Princeton University [2]Princeton University and Institute for Advanced Study. Correspondence to: Yangsibo Huang <yangsibo@princeton.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]https://github.com/Hazelsuko07/InstaHide_Challenge.

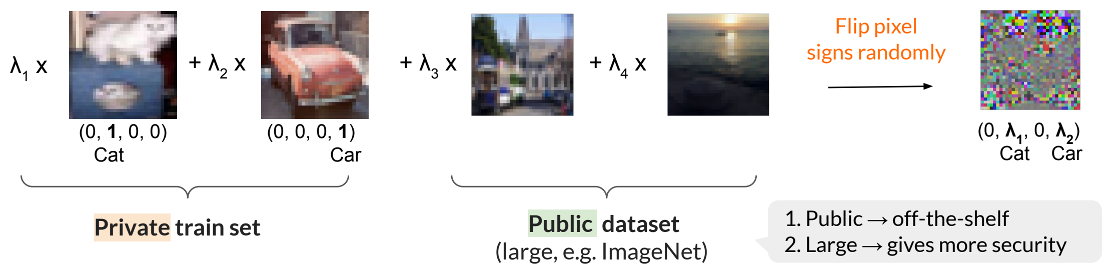


Figure 1: Applying *InstaHide* ($k = 4$) to the leftmost private image consists of mixing with another (private) image randomly chosen from the training set and two images randomly chosen from a fixed large public dataset. This is followed by a random sign-flipping mask on the composite image. To encrypt another image, different random choices will get used ("one-time key").

2018), which trains deep nets on composite images created via linear combination of pairs of images (viewed as vectors of pixel values). In *InstaHide* the first step when encrypting image x (see Figure 1) is to take its linear combination with $k-1$ randomly chosen images from either the participant's private training set or from a large public dataset (e.g., ImageNet (Deng et al., 2009)). The second step of *InstaHide* involves applying a random pattern of sign flips on the pixel values of this composite image, yielding encrypted image $\widetilde{x}$, which can be used as-is in existing deep learning frameworks. Since the set of random images for mixing and the random sign flipped mask are used only once, together as a one-time key, and never used for another encryption.

The idea of random sign flipping[2] is inspired by classic *Instance-Hiding* over finite field GF(2), which involves adding a random vector r to an input x. (See Appendix B for background.) Adding 1 over GF(2) is analogous to a sign flip over $\mathbb{R}$. (Specifically, the groups $(\mathsf{GF}(2), +)$ and $(\{\pm 1\}, \times)$ are isomorphic.) The use of a public dataset in *InstaHide* plays a role reminiscent of *random oracle* in cryptographic schemes (Canetti et al., 2004) —the larger this dataset, the better the conjectured security level (see Section 4). A large private dataset would suffice too for security, but then would require prior coordination/sharing among participants.

Experiments on MNIST, CIFAR-10, CIFAR-100 and ImageNet datasets (see Section 5) suggest that *InstaHide* is an effective approach to hide training images from attackers. It is much more effective at hiding images than *Mixup* alone and provides better trade-off between privacy preservation and accuracy than DP. To enable further rigorous study of attacks, we release a challenge dataset of images encrypted using *InstaHide*.

Enhanced functionality due to *InstaHide*. As hinted above, *InstaHide* plugs seamlessly into existing distributed learning frameworks such as federated learning: clients encrypt their inputs on the fly with *InstaHide* and participate in training (without using DP). Depending upon the level of security needed in the application (see Section 4.1), *InstaHide* can also be used to present enhanced functionality that are unsafe in current distributed frameworks. For instance, in each epoch, computationally limited clients can encrypt each private input x to $\widetilde{x}$ and ship it to the central server for all subsequent computation. The server may randomize in the pooled data to create its own batches to deal with special learning situations when distributed data are not independent and identically distributed.

Rest of the Paper: Section 2 recaps *Mixup* and suggests it alone is not secure. Section 3 presents two *InstaHide* schemes, and Section 4 analyzes their security and provides suggestions for practical use. Section 5 shows experiments for *InstaHide*'s efficiency, efficacy, and security. We review related work in Section 7 and conclude in Section 8.

2. *Mixup* and Its Vulnerability

This section reviews *Mixup* method (Zhang et al., 2018) for data augmentation in deep learning and shows —using two plausible attacks— that it alone does not assure privacy. For ease of description, we consider a vision task with a private dataset $\mathcal{X} \subset \mathbb{R}^d$ of size n. Each image $x_i \in \mathcal{X}, i \in [n]$ is normalized such that $\sum_{j=1}^{d} x_{i,j} = 0$ and $\|x_i\|_2 = 1$.

See Algorithm 1. Given an original dataset, in each epoch of the training the algorithm generates a *Mixup* dataset on the fly, by linearly combining k random samples, as well as their labels (lines 9, 10). *Mixup* suggests that training with mixed samples and mixed labels serves the purpose of data augmentation, and achieves better test accuracy on normal images.

We describe the algorithm as an operation on k images, but previous works mostly used $k = 2$.

[2]Note that randomly flipping signs of coordinates in vector x can be viewed alternatively as retaining only absolute value of each pixel in the mixed image. In other words, for each pixel c in the original image, we scale it by λ and add some value η to it which is the pixel value in images it is mixed with, and we are retaining $|\lambda c + \eta|$. Figure 5 gives an illustration.

We choose to take the viewpoint of random sign flips because this is more useful in extensions of *InstaHide*, which are forthcoming.

Algorithm 1 *Mixup* (Zhang et al., 2018)

```
1:  procedure MIXUP($W, T, \mathcal{X}, \mathcal{Y}$)
2:      $W$: the weights of the deep neural network; T: number of epochs; $\mathcal{X} = \{x_1, \cdots, x_n\}, \mathcal{Y} = \{y_1, \cdots, y_n\}$: the original dataset.
3:      Initialize $W$
4:      for $t = 1 \to T$ do
5:          Generate $\pi_1$ such that $\pi_1(i) = i, \forall i \in [n]$, and $k-1$ random permutations $\pi_2, \cdots, \pi_k : [n] \to [n]$    ▷ $[n]$ denotes $\{1, 2, \cdots, n\}$
6:          Sample $\lambda_1, \cdots, \lambda_n \sim [0,1]^k$ uniformly at random, and for all $i \in [n]$ normalize $\lambda_i$ such that $\|\lambda_i\|_1 = 1$.
7:          $\widetilde{D} \leftarrow \emptyset$
8:          for $i = 1 \to n$ do                ▷ Generate Mixup dataset
9:              $\widetilde{x}_i^{\mathrm{mix}} \leftarrow \sum_{j=1}^k (\lambda_{\pi_j(i)})_j x_{\pi_j(i)}$    ▷ Mix images
10:             $\widetilde{y}_i \leftarrow \sum_{j=1}^k (\lambda_{\pi_j(i)})_j y_{\pi_j(i)}$    ▷ Mix labels
11:             $\widetilde{D} \leftarrow \widetilde{D} \cup (\widetilde{x}_i, \widetilde{y}_i)$
12:         end for
13:         Train $W$ using the Mixup dataset $\widetilde{D}$
14:     end for
15: end procedure
```

2.1. Attack on *Mixup* within a Private Dataset

We propose an attack to the *Mixup* method when a private image is mixed up more than once during training. In fact, in Algorithm 1, each image is used kT times during training, where k is the number of samples to mix, and T is the number of training epochs.

Assume that pairs of images in $\mathcal{X}$ are fairly independent at the pixel level, so that the inner product of a random pair of images (viewed as vectors) has expectation 0 (expectation can be nonzero but small). We can think of each pixel is generated from some distribution with standard deviation $1/\sqrt{d}$ and describe attacks with this assumption. (Section 5 shows that these attacks do work in practice.)

Suppose we have two *Mixup* images $\widetilde{x}_1$ and $\widetilde{x}_2$ which are derived from two subsets of private images, $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{X}$ and $|\mathcal{S}_1| = |\mathcal{S}_2| = k$. If $\widetilde{x}_1$ and $\widetilde{x}_2$ contain different private images, namely $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, then the expectation of $\widetilde{x}_1 \cdot \widetilde{x}_2$ is 0. However, if $\mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset$, and $\widetilde{x}_1$ and $\widetilde{x}_2$ have coefficients λ_1 and λ_2 for the common image in these two sets, then the expectation of $\langle \widetilde{x}_1, \widetilde{x}_2 \rangle$ is $\lambda_1 \lambda_2 / k$, which means by simply checking the inner products between two $\widetilde{x}$'s, the attacker can determine with high probability whether they are derived from the same image. Thus if the attacker finds multiple such pairs, they can average the $\widetilde{x}$'s to start getting a good estimate of x. (Note that the rest of images in the pairs are with high probability distinct and so average to 0.)

2.2. Attack on *Mixup* Between a Private and a Public Dataset

To defend against the previous attack, it seems that a possible method is to modify the *Mixup* method to mix a private

Setting	Secure?
x is mixed in multiple $\widetilde{x}$'s	No
x is mixed in a single $\widetilde{x}$, with a public dataset	No
x is mixed in a single $\widetilde{x}$, with a private dataset	Maybe

Table 1: Security of *Mixup* alone.

image x with $k-1$ images only once to get a single $\widetilde{x}$, and use this $\widetilde{x}$ as surrogate for x in all epochs. To ensure $x \in \mathcal{X}$ is used only once, it uses an additional public dataset $\mathcal{X}'$ (e.g. ImageNet). In other words, for every $x \in \mathcal{X}$, it produces $\widetilde{x}$ by using *Mixup* between x and $k-1$ random images from a large public dataset.

This extension of the *Mixup* method seems secure at first glance, as naively one can imagine that to violate privacy, the adversary must do exhaustive search over $(k-1)$-tuples of public images to determine which were mixed into $\widetilde{x}$, and try all possible k-tuples of coefficients, and then subtract the corresponding sum from $\widetilde{x}$ to extract x. If this is true, it would suggest that extracting x or any approximation to it requires $\binom{N}{k-1} \approx N^{k-1}$ work, where N is the number of images in the public dataset. This work becomes infeasible even for $k = 4$. However, we sketch an attack below that runs in $O(Nk)$ time.

It again uses the above assumption about the pairwise independence property of a random image pair. Recall that standard deviation of pixels is $1/\sqrt{d}$. Namely, to determine the images that went into the mixed sample $\widetilde{x} = \lambda_1 x_1 + \sum_{i=2}^k \lambda_i x_i$, it suffices to go through each image z in the dataset and examine the inner product $z \cdot \widetilde{x}$. If z is not one of the x_i's then this inner product is of the order at most $\sqrt{k/d}$ (see part 1 of Theorem C.3), whereas if it is one of the x_i's then it is of the order at least $\frac{1}{k}(1 - \sqrt{k/d})$ (see part 2 of Theorem C.3). Thus if $k^3 \ll d$ (which is true if the number of pixels d is a few thousand) then the inner product gives a strong signal whether z is one of the x_i's. Once the correct x_i's and their coefficients have been guessed, we obtain x up to a linear scaling. Thus the above attack works with good probability and in time proportional to the size of the dataset. We provide results for this attack in Section 5.

This attack of course requires that x is being mixed in with images from a public dataset $\mathcal{X}'$. In the case that $\mathcal{X}'$ is private and diverse enough, it is conceivable that *Mixup* is safe. (MNIST for example may not be diverse enough but ImageNet probably is.) We leave it as an open question.

2.3. Discussions

Table 1 summarizes the security of *Mixup*. Only *Mixup* with single $\widetilde{x}$ for each private x and within a private dataset(s) has not been identified vulnerable to potential attacks. But this does not necessarily mean it is secure. In addition, the test accuracy in this case is not comparable to vanilla training; it incurs about 20% accuracy loss with CIFAR-10 tasks.

Potentially insecure applications of *Mixup*. Recently, a method called FaceMix (Liu et al., 2019) applies *Mixup* at the representation level (i.e. intermediate output of deep models) during inference. Given a representation function $h : \mathbb{R}^d \to \mathbb{R}^l$ and a secret $x \in \mathbb{R}^d$, the paper assumed a threat model that the attacker is able to reconstruct x given $h(x)$. Therefore, FaceMix proposed to protect the privacy of x by generating $\widetilde{h}_1(x) = \lambda_1 h(x) + \sum_{i=2}^{k} \lambda_i h(x_i)$, and run inference on $\widetilde{h}(x)$. A similar but different idea was shown in (Fu et al., 2019), which proposed to generate $\widetilde{x} = \lambda_1 x + \sum_{i=2}^{k} \lambda_i x_i$ and use $\widetilde{h}_2(x) = h(\widetilde{x})$ for training.

Both methods may be vulnerable to the attacks presented in this section: when h is linear (one example in (Liu et al., 2019)), we have $\widetilde{h}_1(x) = \widetilde{h}_2(x)$, which means the attacker can reconstruct the *Mixup* image $\widetilde{x} = \lambda_1 x + \sum_{i=1}^{k} x_i$ from $\widetilde{h}_1(x)$ and run attacks on *Mixup*. For a nonlinear h, $\widetilde{h}_1(x) \approx \widetilde{h}_2(x)$ may also hold.

3. *InstaHide*

This section first presents two schemes: Inside-dataset *InstaHide* and Cross-dataset *InstaHide*, and then describes their inference.

The Inside-dataset *InstaHide* mixes each training image with random images within the same private training dataset. The Cross-dataset *InstaHide*, arguably more secure (see Section 4), involves mixing with random images from a large public dataset like ImageNet.

3.1. Inside-Dataset *InstaHide*

Algorithm 2 shows Inside-dataset *InstaHide*. Its encryption step includes mixing the secret image x with $k-1$ other training images followed by an extra random pixel-wise sign-flipping mask on the composite image. Note that random sign flipping changes the color of a pixel. The motivation for random sign flips was described around Footnote 2.

Definition 3.1 (random mask distribution $\Lambda_{\pm}^d$)**.** *Let $\Lambda_{\pm}^d$ denote the d-dimensional random sign distribution such that $\forall \sigma \sim \Lambda_{\pm}^d$, for $i \in [d]$, σ_i is independently chosen from $\{\pm 1\}$ with probability 1/2 each.*

To encrypt a private input x_i, we first determine the random coefficient λ's for image-wise combination, but with the constraint that they are at most c to avoid dominant leakage of any single image (line 6 in Algorithm 2). Then we sample a random mask $\sigma_i \sim \Lambda_{\pm}^d$ and apply $\sigma \circ x$, where $\circ$ is coordinate-wise multiplication of vectors (line 10 in Algorithm 2). Note that the random mask σ_i and the $k-1$ images used for mixing with x_i, will not be reused to encrypt other images. They constitute a "random one-time private key."

Algorithm 2 Inside-dataset *InstaHide*

```
1:  procedure InstaHide($W, T, \mathcal{X}, \mathcal{Y}$)                         ▷ This paper
2:      $W$: weights of the neural network; T: number of epochs;
        $\mathcal{X} = \{x_1, \cdots, x_n\}$: data; $\mathcal{Y} = \{y_1, \cdots, y_n\}$: labels.
3:      Initialize $W$
4:      for $t = 1 \to T$ do
5:          Generate $\pi_1$ such that $\pi_1(i) = i, \forall i \in [n]$, and $k-1$
            random permutations $\pi_2, \cdots, \pi_k : [n] \to [n]$
6:          Sample $\lambda_1, \cdots, \lambda_n \sim [0,1]^k$ uniformly at random,
            and for all $i \in [n]$ normalize $\lambda_i \in \mathbb{R}^k$ such that $\|\lambda_i\|_1 = 1$
            and $\|\lambda_i\|_\infty \leq c$.  ▷ $c \in [0,1]$ is a constant that upper bounds
            a single coefficient
7:          Sample $\sigma_1, \cdots, \sigma_n \sim \Lambda_{\pm}^d$ uniformly at random.   ▷
            Definition 3.1
8:          $\widetilde{D} \leftarrow \emptyset$
9:          for $i = 1 \to n$ do                          ▷ Generate InstaHide dataset
10:             $\widetilde{x}_i \leftarrow \sigma_i \circ \sum_{j=1}^{k} (\lambda_{\pi_j(i)})_j x_{\pi_j(i)}$      ▷ Encryption
11:             $\widetilde{y}_i \leftarrow \sum_{j=1}^{k} (\lambda_{\pi_j(i)})_j y_{\pi_j(i)}$             ▷ Mix labels
12:             $\widetilde{D} \leftarrow \widetilde{D} \cup (\widetilde{x}_i, \widetilde{y}_i)$
13:         end for
14:         Train $W$ using the InstaHide dataset $\widetilde{D}$
15:     end for
16: end procedure
```

A priori. It may seem that using a different mask for each training sample would completely destroy the accuracy of the trained net, but as we will see later it has only a small effect when k is small. Mathematically, this seems reminiscent of the *phase retrieval problem* (Candes et al., 2013; Li & Nakos, 2018) (see Appendix E).

3.2. Cross-Dataset *InstaHide*

Cross-Dataset *InstaHide* extends the encryption step of Algorithm 2 by mixing k images from the private training dataset D_{private} and a public dataset D_{public}, and a random mask as a random one-time secret key.

Although the second dataset can be private, there are several motivations to use a public dataset: (a) Some privacy-sensitive datasets, (e.g. CT or MRI scans), feature images with certain structure patterns with uniform backgrounds. Mixing among such images as in Algorithm 2 would not hide information effectively. (b) Drawing mixing images from a larger dataset gives greater unpredictability, hence better security (see Section 4). (c) Public datasets are freely available and eliminate the need for special setups among participants in a distributed learning setting.

To mix k images in the encryption step, we randomly choose $\lceil k/2 \rceil$ images from D_{private} and the other $\lfloor k/2 \rfloor$ from D_{public}, and apply *InstaHide* to all these images. The only difference in the Cross-dataset scheme is that, the model is trained to learn *only* the (mixed) label of D_{private} images. We assume D_{public} images are unlabelled.

We advocate preprocessing a public dataset in two steps to obtain D_{public} for better security. The first is to randomly crop a number of patches from each image in the public dataset to form D_{public}. This step will make D_{public} much larger than the original public dataset. The second is to filter out the "flat" patches. In our implementation, we design a filter using SIFT (Lowe, 1999), a feature extraction technique to retain patches with more than 40 key points.

3.3. Inference with *InstaHide*

Either scheme above by default applies *InstaHide* during inference, by averaging predictions of multiple encryptions (e.g. 10) of a test sample. This idea is akin to existing cryptographic frameworks for secure evaluation on a public server via homomorphic encryption (e.g. (Mishra et al., 2020)). Since the encryption step of *InstaHide* is very efficient, the overhead of such inference is quite small.

One can also choose not to apply *InstaHide* during inference. We found in our experiments (Section 5) that it works for low-resolution image datasets such as CIFAR-10 but it does not work well with a high-resolution image dataset such as ImageNet.

4. Security Analysis

This section considers the security of *InstaHide* in distributed learning, specifically the Cross-dataset version.

Attack scenario: In each epoch, all clients replace each (image, label)-pair (x, y) in the training set with some $(\widetilde{x}, \widetilde{y})$ using *InstaHide*. Attackers observe $h(\widetilde{x}, \widetilde{y})$ for some function h: in federated learning h could involve batch gradients or hidden-layer activations computed using input $\widetilde{x}, \widetilde{y}$ as well as other inputs.

Argument for security consists of two halves: (1) To recover significant information about an image x from communicated information, *computationally limited* eavesdroppers/attackers have to break *InstaHide* encryption (Section 4.1). (2) Breaking *InstaHide* is difficult (Section 4.2).

4.1. Secure Encryption Implies Secure Protocol

Suppose an attacker exists that compromises an image x in the protocol. We do the thought experiment of even providing the attacker with encryptions of *all* images belonging to all parties, as well as model parameters in each iteration. Now everything the attacker sees during the protocol it can efficiently compute by itself, and we can convert the attacker to one that, given $\widetilde{x}$, extracts information about x. We conclude in this thought experiment that a successful attack on the protocol also yields a successful attack on the encryption. In other words, privacy loss during protocol is *upper bounded* by privacy loss due to the encryption itself. Of course, this proof allows the possibility that the protocol —due to aggregation of gradients, etc.—ensures even greater privacy than the encryption alone.

Dealing with multiple encryptions of same x: In our protocol each image x is re-encrypted in each epoch. This seems to act as a data augmentation and improves accuracy. The above argument translated to this setting shows that privacy violation requires solving the following problem: Private images $x_1, x_2, \ldots, x_m$ were each encrypted T times (m is size of the private training set, T is number of epochs), each time using a new private key. Attacker is given these mT encryptions. *Weaker task:* Attacker has to identify which of them came from x_1. *Stronger task:* Attacker has to identify x_1.

We conjecture both tasks are hard. Visualization (see Figure 8) as well as the Kolmogorov–Smirnov test (Kolmogorov, 1933; Smirnov, 1948) (see Appendix F) suggest that statistically, it is difficult to distinguish among distributions of encryptions of different images. Thus, effectively identifying multiple $\widetilde{x}$'s of same x and using them to run attacks seems difficult.

4.2. Hardness of Attacking *InstaHide* Encryption

Now we consider the difficulty of recovering information about x given a single encryption $\widetilde{x}$.

Security estimates of naive attack. We start by considering the naive attack, which would involve the attacker to either figure out the set of all k images, or to compromise the mask σ and run attacks on *Mixup*. This should take $\min\{|\mathcal{X} \cup \mathcal{X}'|^{k/2}, 2^d\}$ time. For inside-dataset *InstaHide* (i.e., $\mathcal{X}' = \emptyset$), the size of the private dataset $\mathcal{X}$ is usually of order 10^5, thus the attack takes $10^{2.5k}$. For cross-dataset *InstaHide* with a large public dataset (e.g. ImageNet), the computation cost of attack will increase to $10^{3.5k}$. For both cases, $k = 4$ already makes the attack hard.

Now we suggest reasons why the naive attack may be best possible.

For worst-case pixel-vectors, finding the k-image set is hard. Appendix F shows that for *worst-case* choices of images (i.e., when an "image" is allowed to be an arbitrary sequence of pixel values) the computational complexity of this problem is related to the famous k-VECTOR SUBSET SUM problem[3], whose complexity is conjectured to be $|\mathcal{X} \cup \mathcal{X}'|^{k/2}$ under a strong form of the P vs NP conjecture. Thus when $\mathcal{X}'$ is a large public dataset and $k \geq 4$, the computational effort to recover the image ought to be at least quadratic in the dataset size, which should be of the order of 10^{10} or more.

[3]Given a set of N public vectors $v_1, \cdots v_N \in \mathbb{R}^d$ and $\sum_j^k v_{i_j}$, the sum of a secret subset $i_1, \cdots, i_k$ of size k, k-VECTOR SUBSET SUM aims to find $i_1, \cdots, i_k$.

Of course, images are not worst-case vectors. Thus an attack must leverage this fact somehow. The obvious idea today is to use a deep net for the attack, and the experiments below will suggest the obvious ideas do not work.

Compromising the mask is also hard. As previously discussed, the pixel-wise mask $\sigma \sim \Lambda_{\pm}^{d}$ (Def. 3.1) in *InstaHide* is kept private by each client, which is analogous to a private key (assuming the client never shares its own σ with others, and the generation of σ is statistically random). Brute-force algorithm consumes 2^d time to figure out σ, where d can be several thousands in vision tasks.

5. Experiments

We have conducted experiments to answer three questions:

1. How much accuracy loss does *InstaHide* suffer (Section 5.1)?
2. How is the accuracy loss of *InstaHide* compared to differential privacy approaches (Section 5.2)?
3. Can *InstaHide* defend against known attacks (Section 5.3)?

We are particularly interested in the cases where $k = 4$.

Datasets and setup. Our main experiments are image classification tasks on four datasets MNIST (LeCun et al., 2010), CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and ImageNet (Deng et al., 2009). We use ResNet-18 (He et al., 2016) architecture for MNIST and CIFAR-10, NasNet (Zoph et al., 2018) for CIFAR-100, and ResNeXt-50 (Xie et al., 2017) for ImageNet. The implementation uses the Pytorch (Paszke et al., 2019) framework. Note that we convert greyscale MNIST images to be 3-channel RGB images in our experiments. Hyper-parameters are provided in Appendix F.

5.1. Accuracy Results of *InstaHide*

We evaluate the following *InstaHide* variants:

- Inside-dataset *InstaHide* with different k's, where k is chosen from $\{1, 2, 3, 4, 5, 6\}$.
- Cross-dataset *InstaHide* with $k = 4$. For MNIST, we use CIFAR-10 as the public dataset; for CIFAR-10 and CIFAR-100, we use the resized ImageNet as the public dataset. We do not test Cross-dataset *InstaHide* on ImageNet since its sample size is already large enough for good security.

The computation overhead of *InstaHide* in terms of extra training time in our experiments is smaller than 5%.

Accuracy with different k's. Figure 2 shows the test accuracy of vanilla training, and inside-dataset *InstaHide* with different k's on MNIST, CIFAR-10 and CIFAR-100 benchmarks. Compared with vanilla training, *InstaHide* with $k = 4$ only suffers small accuracy loss of 1.3%, 3.4%, and 4.7% respectively. Also, increasing k from 1 (i.e., apply mask on original images, no *Mixup*) to 2 (i.e., apply mask on pairwise mixed images) improves the test accuracy for CIFAR datasets, suggesting that *Mixup* data augmentation also helps for encrypted *InstaHide* data points.

Inside-dataset v.s. Cross-dataset. We also evaluate the performance of Cross-dataset *InstaHide*, which does encryption using random images from both the private dataset and a large public dataset. As shown in Table 2, Cross-dataset *InstaHide* incurs an additional small accuracy loss comparing with inside-dataset *InstaHide*. The total accuracy losses for the three cases are 1.4%, 5.5%, and 5.8% respectively. As previously suggested, a large public dataset provides a stronger notion of security.

Inference with and without *InstaHide*. As mentioned in Sec 3.3, by default, *InstaHide* is applied during inference. In our experiments, we averaging predictions of 10 encryptions of a test image. We found that for high-resolution images, applying *InstaHide* during inference is important. The results of using Inside-dataset *InstaHide* on ImageNet in Table 2 show that the accuracy of inference with *InstaHide* is 72.6%, whereas that without *InstaHide* is only 1.4%.

5.2. *InstaHide* vs. Differential Privacy approaches

Although *InstaHide* is qualitatively different from differential privacy in terms of privacy guarantee, we would like to provide hints for their relative accuracy (Question 2).

Comparison with DPSGD. DPSGD (Abadi et al., 2016) injects noise to gradients to control private leakage. Table 2 shows that DPSGD leads to an accuracy drop of about 20% on CIFAR-10 dataset. By contrast, *InstaHide* gives models almost as good as vanilla training in terms of test accuracy.

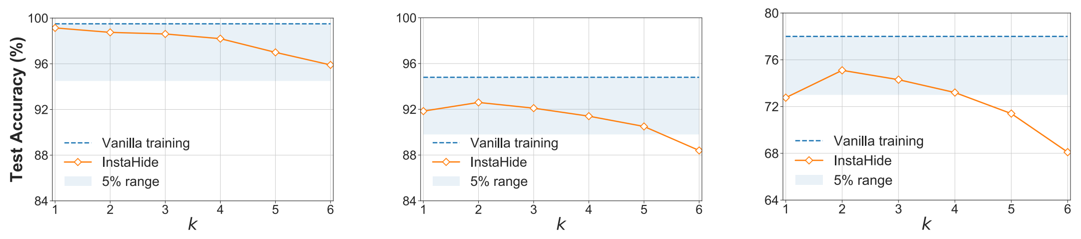
Comparison with adding random noise to images. We also compare *InstaHide* (i.e., adding *structured* noise) with adding random noise to images (another typical approach to preserve differential privacy). Specifically, given the original dataset $\mathcal{X}$, and a perturbation coefficient $\alpha \in (0, 1)$, we test a) *InstaHide*$_{\text{inside},k=2}$: $\widetilde{x}_i = \sigma \circ ((1-\alpha)x_i + \alpha x_j)$, where $x_j \in \mathcal{X}$, $j \neq i$, and b) adding random Laplace noise e: $\widetilde{x}_i = (1-\alpha)x_i + e$, where $\mathbb{E}[\|e\|_1] = \mathbb{E}[\|\alpha x_j\|_1]$.

As shown in Figure 3, by increasing α from 0.1 to 0.9, the test accuracy of adding random noise drops from $\sim$ 94% to $\sim$ 10%, while the accuracy of *InstaHide* is above 90%.

5.3. Vulnerability Against Attacks

To answer the question how well *InstaHide* can defend known attacks, here we report our findings with a sequence of attacks on *InstaHide* encryption $\widetilde{x}$ to recover original image x, including gradient-matching attack (Zhu et al., 2019), demasking using GAN (Generative Adversarial Network), averaging multiple encryptions, and uncovering public images with similarity search.

Gradient-matching attack (Zhu et al., 2019). Here attacker observes gradients of the loss generated using a



(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Figure 2: Test accuracy $(\%)$ on MNIST, CIFAR-10 and CIFAR-100 for vanilla training and inside-dataset *InstaHide* with different k's. *InstaHide* with $k \leq 4$ only suffers small accuracy loss.

	MNIST	**CIFAR-10**	**CIFAR-100**	**ImageNet**	***Assumptions***
Vanilla training	99.5 ± 0.1	94.8 ± 0.1	77.9 ± 0.2	77.4	-
DPSGD*	98.1	72.0	N/A	N/A	*A **labeled** public dataset for pre-training*
InstaHide$_{\text{inside},k=4,\text{ in inference}}$	98.2 ± 0.2	91.4 ± 0.2	73.2 ± 0.2	72.6	-
InstaHide$_{\text{inside},k=4}$	98.2 ± 0.3	91.2 ± 0.2	73.1 ± 0.3	1.4	-
InstaHide$_{\text{cross},k=4,\text{ in inference}}$	98.1 ± 0.2	89.3 ± 0.2	72.1 ± 0.3	-	*Using a large **unlabelled** public dataset*
InstaHide$_{\text{cross},k=4}$	98.0 ± 0.2	89.2 ± 0.3	72.1 ± 0.3	-	

Table 2: Test accuracy $(\%)$ on MNIST, CIFAR-10, CIFAR-100 and ImageNet for vanilla training, DPSGD (Abadi et al., 2016) and *InstaHide*, including the mean and standard deviation of test accuracy across 5 runs except for ImageNet. *DPSGD results are from (Papernot et al., 2019), which does not have results for CIFAR-100 and Imagenet. Results marked with "in inference" applies *InstaHide* during inference. *InstaHide* methods incur minor accuracy reductions.

user's private image s while training a deep net (attacker knows the deep net, e.g., as a participant in Federated Learning) and tries to recover s by computing an image s^* that has similar gradients to those of s (see algorithm in Appendix F). Figure 4 shows results of this attack on *Mixup* and *InstaHide* schemes on CIFAR-10. If *Mixup* with $k = 4$ is used, the attacker can still extract fair bit of information about the original image. However, if *InstaHide* is used the attack isn't successful.

Demask using GAN. *InstaHide* does pixel-wise random sign-flip after applying *Mixup* (with public images, in the most secure version). This flips the signs of half the pixels in the mixed image. An alternative way to think about it is that the adversary sees the intensity information (i.e. absolute value) but not the sign of the pixel. Attackers could use computer vision ideas to recover the sign. One attack consists of training a GAN on this sign-recovery task[4], using a large training set of $(z, \sigma \circ z)$ where z is a mixed image and σ is a random mask. If this GAN recovers the signs reliably, this effectively removes the mask, after which one could use the attacks against *Mixup* described in Section 2.

In experiments this only succeeded in recovering half the flipped signs, which means $\sim 1/4$ of the coordinates continued to have the wrong sign; see Figure 5. GAN training[5] used 10,000 cross-dataset *InstaHide* examples generated with CIFAR-10 and ImageNet and $k = 4$. This level of sign recovery seems insufficient to allow the attack against *Mixup* (Section 2) to succeed, nor the other attacks discussed below. Nevertheless, researchers trying to break *InstaHide* may want to use such a demasking GAN as a starting point.

Average multiple encryptions of the same image. We further test if different encryptions of the same image (after demasking) can be used to recover that hidden image by running the attack in Section 2.1.

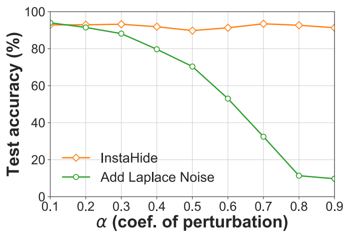
Assuming a public history of $n \times T$ encryptions, where n is the size of the private training set, and T is the number of epochs. We consider a stronger and a weaker version of this attack.

- **Stronger attack**: the attacker already *knows* the set of multiple encryptions of the same image x. He uses GAN to demask all encryptions in the set, and averages images in the demasked set to estimate x.
- **Weaker attack**: the attacker *does not know* which subset of the encryption history correspond to the same original image. To identify that subset, he firstly demasks all $n \times T$ encrytions in the history using GAN. With an arbitrary demasked encryption (from the history) for some unknown original image x, he runs similarity search to find top-m closest images in $n \times T - 1$ other demasked encryptions (which may also contain x), and averages these $m + 1$ images to estimate x.

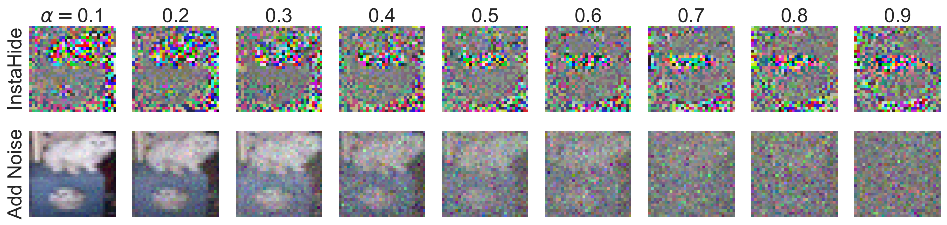
The stronger attack is conceivable if n is very small (say a hospital only has 100 images), so via brute force the attacker can effectively have a small set of encryptions of the

[4]We thank Florian Tramèr for suggesting this attack.

[5]We use this GAN architecture (designed for image colorization): https://github.com/zeruniverse/neural-colorization.

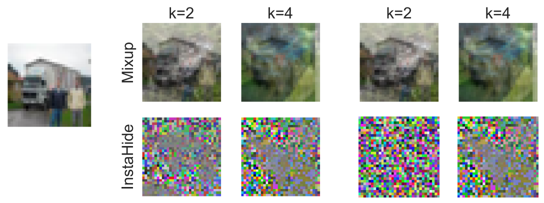


(a) Accuracy with different α's.



(b) Images generated by *InstaHide* and adding Laplace noise with different α's.

Figure 3: Relative accuracy on CIFAR-10 (a) and visualization (b) of *InstaHide* and adding random Laplace noise with different α's, the coefficient of perturbation. *InstaHide* gives better test accuracy than adding random noise.



(a) Original (b) After mixing (c) Attack results

Figure 4: Visualization of (a) the original image, (b) *Mixup* and *InstaHide* images, and (c) images recovered by the *gradients matching attack*. *InstaHide* is more effective in hiding the image than *Mixup*.

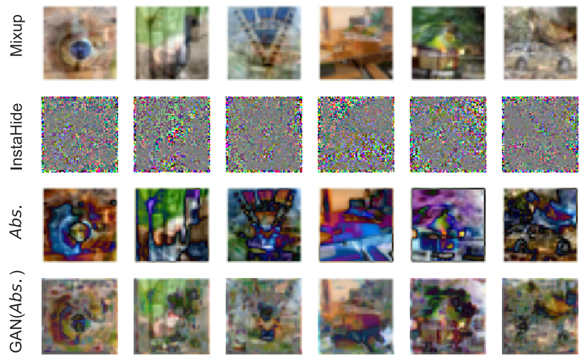


Figure 5: Undo sign-flipping using GAN. Rows: (1) Output from *Mixup* algorithm (Algorithm 1, line 9); (2) Result of applying random mask σ on previous row (Algorithm 2, line 10). Note that randomly flipping sign of a pixel still preserves its absolute value. (3) Taking coordinate-wise absolute value of previous row. (4) Output of demasking GAN on previous row. The attack corrects about $1/4$ of the flipped signs but this doesn't appear enough to allow further attacks that recover the encrypted image.

same image. However, in practice, n is usually at least a few thousand.

For simplicity, we test with $n = 50$ and $T = 50$ (a larger n will make the attack harder). We use the structural similarity index measure (SSIM) (Wang et al., 2004) as the similarity metric, and set m to 5 after tuning.

We also run this attack directly on *Mixup* for comparison.

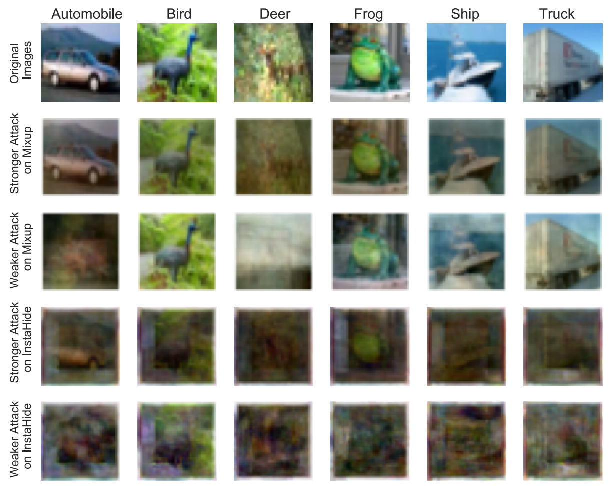


Figure 6: Average multiple encryptions (after demasking) of the same image to attack ($k = 4$). In the stronger attack, the attacker already knows the set of multiple encryptions of the same image; in the weaker attack, the attacker has to identify that set first. Rows: (1) Original images. (2-3) Results of the stronger and the weaker attacks on *Mixup*. (4-5) Results of the stronger and the weaker attacks on *InstaHide*. Note that the attacker has to demask *InstaHide* encryptions using GAN before running attacks, and the information loss of this step makes it harder to attack *InstaHide* than the plain *Mixup*.

As shown in Figure 6, if the original image is not flat (e.g. the "deer"), the stronger attack may not work. For flat images (e.g. the "truck") or images with strong contrast (e.g. the "automobile" and the "frog"), the stronger attack is able to vaguely recover the original image. However, as previously suggested, the stronger attack is conceivable for a very small n.

Note that results here upper bound the privacy leakage in real-world scenarios since we assume a perfect recovery of $\widetilde{x}$ from the gradients.

Uncover public images by similarity search. We also run the attack in Section 2.2 after demasking *InstaHide* encryptions using GAN, which tries to uncover the public images for mixing by running similarity search in the public

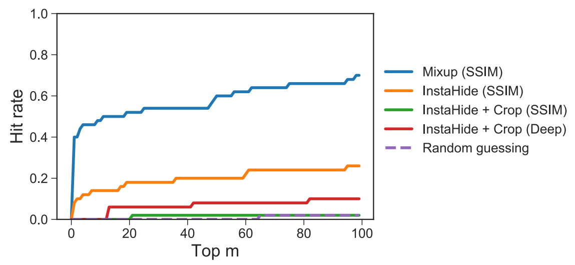


Figure 7: Averaged hit rate of uncovering public images for mixing among the top-m answers returned by similarity search. Running this attack on *InstaHide* requires GAN-demasking as the first step, and a wrongly demasked pixel will make the similarity score less reliable and yield lower hit rate. Mixing with random cropped patches of public images augments the public dataset and gives more security. It also disables similarity search using SSIM. Train a deep model to predict similairty score also does not give a promising hit rate.

dataset using the demasked encryption as the query.

We test with $k = 4$: mix 2 private images from CIFAR-10 with 2 public images from a set of 10,000 ImageNet images (i.e. $N = 10,000$). We consider the attack a 'hit' if at least one public image for mixing is among the top-m answers of the similarity search. The attacker uses SSIM as the default similarity metric for search. However, a traditional alignment-based similarity metric (e.g SSIM) would fail in *InstaHide* schemes which use randomly cropped patches of public images for mixing (see Figure 7), so in that case, the attacker trains a deep net (VGG (Simonyan & Zisserman, 2015) in our experiments) to predict the similarity score.

Note that to find the 2 correct public images for mixing, the attacker has to try all $\binom{m}{2}$ combinations of the top-m answers with different coefficients, and subtract the combined image from the demasked *InstaHide* encryption to verify. Figure 7 reports the averaged hit rate of this attack on 50 different *InstaHide* images. As shown, even with a relatively small public dataset ($N = 10,000$) and a large $m = \sqrt{N}$, the hit rate of this attack on *InstaHide* (enhanced with random cropping) is around 0.1 (i.e. the attacker still has to try $\binom{m}{2} = N/2$ combinations to succeed with probability 0.1). Also, it is conceivable that this attack becomes much more expensive with the public dataset being the whole ImageNet dataset ($N = 1.4 \times 10^7$) or random images on the Internet.

6. *InstaHide* Deployment: Best practice

Based on our security analysis (sec 4 and sec 5.3), we suggest the following:

- Consider Inside-dataset *InstaHide* only if the private dataset is very large and images have varied, complex patterns. If the images in the private dataset have simple signal patterns or the dataset size is relatively small, consider using Cross-dataset *InstaHide*.
- For Cross-dataset *InstaHide*, use a very large public dataset. Follow the preprocessing steps advocated in Sec 3.2 to randomly crop patches from each image in the public dataset and filter out "flat" patches.
- Re-encrypt images in each epoch. This allows the benefits of greater data augmentation for deep learning and hinders attacks (as suggested in Sec 5.3).
- Since images are re-encrypted in each epoch, for best security (e.g, against gradient-matching attacks), each participant should perform a random re-batching so that batch gradients do not correspond to the same subset of underlying images.
- Choose $k = 4, 5$ for a good trade-off between accuracy and security.
- Set a conservative upper threshold for the coefficients in mixing (e.g. 0.65 in our experiments).

A challenge dataset. To encourage readers to design stronger attacks, we release a challenge dataset[6] of encrypted images generated by applying Cross-dataset *InstaHide* with $k = 4$ on some private image dataset and a preprocessed ImageNet as the public dataset. An attack is considered to succeed if it substantially recovers a significant fraction of original images.

7. Related Work

See Appendix A.

8. Conclusion

InstaHide is a practical instance-hiding method for image data for private distributed deep learning.

InstaHide uses the *Mixup* method with a one-time secret key consisting of a pixel-wise random sign-flipping mask and samples from the same training dataset (Inside-dataset *InstaHide*) or a large public dataset (Cross-dataset *InstaHide*). The proposed method can be easily plugged into any existing distributed learning pipeline. It is very efficient and incurs minor reduction in accuracy.

We hope our analysis of *InstaHide*'s security on worst-case vectors will motivate further theoretical study, including for average-case settings and for adversarial robustness. In Appendix E, we suggest that although *InstaHide* can be formulated as a phase retrieval problem, classical techniques have failed as attacks.

We have tried statistical and computational attacks against *InstaHide* without success. To encourage other researchers to try new attacks, we release a challenge dataset of encrypted images.

[6]https://github.com/Hazelsuko07/InstaHide_Challenge.

Acknowledgments

This project is supported in part by Princeton University fellowship, Ma Huateng Foundation, Schmidt Foundation, Simons Foundation, NSF, DARPA/SRC, Google and Amazon AWS.

We would like to thank Amir Abboud, Josh Alman, Boaz Barak, and Hongyi Zhang for helpful discussions, and Mark Braverman, Matthew Jagielski, Florian Tramèr for suggesting attacks.

References

Abadi, M., Feigenbaum, J., and Kilian, J. On hiding information from an oracle. In *Proceedings of the nineteenth annual ACM symposium on Theory of Computing (STOC)*, 1987.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016.

Abboud, A. Fine-grained reductions and quantum speedups for dynamic programming. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2019.

Abboud, A. and Lewi, K. Exact weight subgraphs and the k-sum conjecture. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2013.

Abboud, A., Lewi, K., and Williams, R. Losing weight by gaining edges. In *European Symposium on Algorithms (ESA)*, 2014.

Act, A. Health insurance portability and accountability act of 1996. *Public law*, 104:191, 1996.

Aono, Y., Hayashi, T., Wang, L., and Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

Arora, S. and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Baran, I., Demaine, E. D., and Pǎtraşcu, M. Subquadratic algorithms for 3sum. *Algorithmica*, 50(4):584–596, 2008.

Beckham, C., Honari, S., Verma, V., Lamb, A., Ghadiri, F., Hjelm, R. D., Bengio, Y., and Pal, C. On adversarial mixup resynthesis. In *NeurIPS*, 2019.

Beimel, A. Secret-sharing schemes: a survey. In *International conference on coding and cryptology*, pp. 11–46. Springer, 2011.

Bernstein, S. On a modification of chebyshev's inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924.

Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A., and Raffel, C. MixMatch: A holistic approach to semi-supervised learning. In *Conference on Neural Information Processing Systems (NeurIPS)*. `http://arxiv.org/pdf/1905.02249.pdf`, 2019.

Bhattacharyya, A., Indyk, P., Woodruff, D. P., and Xie, N. The complexity of linear dependence problems in vector spaces. In *ICS*, pp. 496–508, 2011.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

Candes, E. J., Romberg, J. K., and Tao, T. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

Candes, E. J., Strohmer, T., and Voroninski, V. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 66(8):1241–1274, 2013.

Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4): 557–594, 2004.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Dolev, S., Gupta, P., Li, Y., Mehrotra, S., and Sharma, S. Privacy-preserving secret shared computations using mapreduce. *IEEE Transactions on Dependable and Secure Computing*, 2019.

Donoho, D. L. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006.

Dwork, C. The differential privacy frontier. In *Theory of Cryptography Conference*, pp. 496–502. Springer, 2009.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 486–503. Springer, 2006.

Erickson, J. Lower bounds for linear satisfiability problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 388–395, 1995.

Fienup, J. R. Reconstruction of an object from the modulus of its fourier transform. *Optics letters*, 3(1):27–29, 1978.

Fienup, J. R. Phase retrieval algorithms: a comparison. *Applied optics*, 21(15):2758–2769, 1982.

Fu, Y., Wang, H., Xu, K., Mi, H., and Wang, Y. Mixup based privacy preserving mixed collaboration learning. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 275–2755, 2019.

Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC)*, pp. 169–178, 2009.

Gilbert, A. C., Li, Y., Porat, E., and Strauss, M. J. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing 2012 (A preliminary version of this paper appears in STOC 2010)*, 41(2):436–453, 2010.

Graepel, T., Lauter, K., and Naehrig, M. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pp. 1–21. Springer, 2012.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 770–778, 2016.

Impagliazzo, R., Paturi, R., and Zane, F. Which problems have strongly exponential complexity? In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 653–662, 1998.

Jiang, S., Song, Z., Weinstein, O., and Zhang, H. Faster dynamic matrix inverse for faster lps. In *arXiv preprint*. https://arxiv.org/pdf/2004.07470, 2020.

Karp, R. M. Reducibility among combinatorial problems. In *Complexity of computer computations*, pp. 85–103. Springer, 1972.

Kolmogorov, A. Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn.*, 4:83–91, 1933.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*. http://arxiv.org/pdf/1610.05492.pdf, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Laurent, B. and Massart, P. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pp. 1302–1338, 2000.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. In *ATT Labs*, volume 2. http://yann.lecun.com/exdb/mnist, 2010.

Li, P., Li, J., Huang, Z., Li, T., Gao, C.-Z., Yiu, S.-M., and Chen, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, 74:76–85, 2017.

Li, Y. and Nakos, V. Sublinear-time algorithms for compressive phase retrieval. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 2301–2305. https://arxiv.org/pdf/1709.02917.pdf, 2018.

Liu, Z., Wu, Z., Zhu, L., Gan, C., and Han, S. Facemix: Privacy-preserving facial attribute classification on the cloud. In *Han Lab Tech report*, 2019.

Lowe, D. G. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pp. 1150–1157. Ieee, 1999.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., and Popa, R. A. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

Mohassel, P. and Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38. IEEE, 2017.

Moravec, M. L., Romberg, J. K., and Baraniuk, R. G. Compressive phase retrieval. In *Wavelets XII*, volume 6701, pp. 670120. International Society for Optics and Photonics, 2007.

Nakos, V. *Sublinear-Time Sparse Recovery, and Its Power in the Design of Exact Algorithms*. PhD thesis, Harvard University, 2019.

Nakos, V. and Song, Z. Stronger l2/l2 compressed sensing; without iterating. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 289–297. https://arxiv.org/pdf/1903.02742.pdf, 2019.

Pang, T., Xu, K., and Zhu, J. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *International Conference on Learning Representations (ICLR)*. http://arxiv.org/pdf/1909.11515, 2019.

Papernot, N., Chien, S., Song, S., Thakurta, A., and Erlingsson, U. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. In *Manuscript*. https://openreview.net/pdf?id=rJg851rYwH, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

Patrascu, M. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing (STOC)*, 2010.

Price, E. C. *Sparse recovery and Fourier sampling*. PhD thesis, Massachusetts Institute of Technology, 2013.

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (CCS)*, pp. 1310–1321. ACM, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/pdf/1409.1556, 2015.

Smirnov, N. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*, 19(2): 279–281, 1948.

Song, Z. *Matrix theory: optimization, concentration, and algorithms*. PhD thesis, The University of Texas at Austin, 2019.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning (ICML)*, pp. 6438–6447, 2019.

Voigt, P. and Von dem Bussche, A. The EU general data protection regulation (GDPR). *Intersoft consulting*, 2018.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

Yao, A. C. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 160–164, 1982.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*. `http://arxiv.org/pdf/1710.09412`, 2018.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 14747–14756. `http://arxiv.org/pdf/1906.08935`, 2019.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 8697–8710, 2018.