What Can Phase Retrieval Tell Us About Private Distributed Learning?

Anonymous authors
Paper under double-blind review

Abstract

In this work, we examine the security of InstaHide, a scheme recently proposed by Huang et al. (2020) for preserving the security of private datasets in the context of distributed learning. To generate a synthetic training example to be shared among the distributed learners, InstaHide takes a convex combination of private feature vectors and randomly flips the sign of each entry of the resulting vector with probability 1/2. A salient question is whether this scheme is secure in any provable sense, perhaps under a plausible complexity-theoretic assumption.

The answer to this turns out to be quite subtle and closely related to the average-case complexity of a new multi-task, missing-data version of the classic problem of phase retrieval that is interesting in its own right. Motivated by this connection, under the standard distributional assumption that the public/private feature vectors are isotropic Gaussian, we design an algorithm that can actually recover a private vector using only the public vectors and synthetic vectors generated by InstaHide.

1 Introduction

In distributed learning, where decentralized parties each possess some private local data and work together to train a global model, a central challenge is to ensure that the security of any individual party's local data is not compromised. Huang et al. (2020) recently proposed an exciting approach called *InstaHide* for this problem. At a high level, InstaHide is a method for aggregating local data into synthetic data that can both preserve the privacy of the local datasets and be used to train good models.

Informally, given a collection of *public feature vectors* (e.g. a publicly available dataset like ImageNet) and a collection of *private feature vectors* (e.g. the union of all of the private datasets among learners), InstaHide produces a synthetic feature vector as follows. Let positive integers $k_{\mathsf{pub}}, k_{\mathsf{priv}}$ be sparsity parameters.

1. Form a random convex combination of k_{pub} public and k_{priv} private vectors.
2. Multiply every coordinate of the resulting vector by an independent random sign in $\{\pm 1\}$, and define this to be the synthetic feature vector.

The intuition from Huang et al. (2020) is that by removing any sign information from the vector obtained in Step 1, Step 2 makes it difficult to discern which public and private vectors were selected in Step 1. Strikingly, Huang et al. (2020) demonstrated on real-world datasets that if one trains a ResNet-18 or a NASNet on a dataset consisting of synthetic feature vectors generated in this fashion, one can still get good *test accuracy* on the constituent private and public datasets for modest sparsity parameters (e.g. $k_{\mathsf{pub}} = k_{\mathsf{priv}} = 2$).[1]

The two outstanding *theoretical* challenges that InstaHide poses are understanding:

[1] We did not describe how the labels for the synthetic feature vectors are assigned, but this part of InstaHide will not be important for our theoretical results and we defer discussion of labels to Section 4.

- **Utility**: What property, either of neural networks or of real-world distributions, lets one tolerate this kind of covariate shift between the synthetic and original datasets?
- **Security**: Under what settings can we *rigorously prove* that InstaHide preserves security of the private datasets used to generate the synthetic vectors?

In this paper we focus on the latter, which we make precise in Section 2.1. Of course, any meaningful notion of security for InstaHide should be sensitive to the properties of the distributions generating the private and public feature vectors.

From a theoretical standpoint, one of the most standard distributional assumptions one can make is that of *Gaussianity*, i.e. that the private and public data are i.i.d. samples from a multivariate normal distribution. While real-world datasets do not actually satisfy this, such an assumption is arguably the most common starting points for works aiming to develop rigorous guarantees for algorithmic problems in supervised and unsupervised learning Moitra & Valiant (2010); Netrapalli et al. (2013); Candes et al. (2015); Hardt & Price (2015); Zhong et al. (2017b;a); Li & Yuan (2017); Ge et al. (2018); Li & Liang (2018); Zhong et al. (2019); Diakonikolas et al. (2020), and a natural first step towards understanding either of the two questions above is to specialize to this case.

Our main result is to show that when the private and public data is Gaussian, then we can use the synthetic dataset and the public feature vectors to recover a private feature vector.

Theorem 1.1 (Informal, see Theorem A.1)**.** *If there are n_{priv} private feature vectors and n_{pub} public feature vectors, each of which is an i.i.d. draw from $\mathcal{N}(0, \mathsf{Id}_d)$, then as long as $d \geq \Omega(\mathrm{poly}(k_{\mathsf{pub}}, k_{\mathsf{priv}}) \log(n_{\mathsf{pub}} + n_{\mathsf{priv}})$, there is some $M = o(n_{\mathsf{priv}}^{k_{\mathsf{priv}}})$ such that, given a sample of M random synthetic feature vectors independently generated as above, one can* exactly recover *$k_{\mathsf{priv}} + 2$ private feature vectors in time $\widetilde{O}(k_{\mathsf{pub}}^2 d) + O(M^2)$ with high probability.*[2]

We emphasize that we can take $M = o(n_{\mathsf{priv}}^{k_{\mathsf{priv}}})$, meaning we can achieve recovery well before we have seen all possible combinations of private vector among the synthetic vectors generated (see Remark A.2 for additional discussion).

Additionally, to ensure we are not working in an uninteresting setting where InstaHide has zero *utility*, we empirically verify that in the setting of Theorem 1.1, one can train on the synthetic vectors and get good test accuracy on the original Gaussian dataset (see Section 4).

Qualitatively, the main takeaway of Theorem 1.1 is that to prove meaningful security guarantees for InstaHide, we must be careful about the properties we posit about the underlying distribution generating the public and private data. In other words, while it could very well be that no efficient algorithm can take the synthetic dataset generated by InstaHide and extract a private feature vector when the public and private datasets are real-world datasets, *any such claim of security will need to depend on the non-Gaussianity of real-world data.*

1.1 Connections and Extensions to Phase Retrieval

Our algorithm is based on connections and extensions to the classic problem of *phase retrieval.* At a high level, this can be thought of as the problem of linear regression where the signs of the linear responses are hidden. More formally, this is a setting where we get pairs $(x_1, y_1), ..., (x_N, y_N) \in \mathbb{C}^n \times \mathbb{R}$ for which there exists a vector $w \in \mathbb{C}^n$ satisfying $|\langle w, x_i \rangle| = y_i$ for all $i = 1, ..., N$, and the goal is to recover w. Without distributional assumptions on how $x_1, ..., x_N$ are generated, this problem is NP-hard Yi et al. (2014), and in the last decade, there has been a huge body of work, much of it coming from the machine learning community, on giving algorithms for recovering w under the assumption that $x_1, ..., x_N$ are i.i.d. Gaussian, see e.g. Candes et al. (2013; 2015); Conca et al. (2015); Netrapalli et al. (2013).

To see the connection between InstaHide and phase retrieval, first imagine that InstaHide only works with public feature vectors (in the notation of Theorem 1.1, $n_{\mathsf{priv}} = k_{\mathsf{priv}} = 0$). Now, consider a synthetic feature vector $y \in \mathbb{R}^d$ generated by InstaHide, and let the vector

[2] See Problem 1 and Remark 2.7 for a discussion of what exact recovery precisely means in this context.

$w \in \mathbb{R}^{n_{\mathsf{priv}}+n_{\mathsf{pub}}}$ be the one specifying the convex combination of public feature vectors that generated y. The basic observation is that for any feature $i \in [d]$, if $p_i \in \mathbb{R}^{n_{\mathsf{priv}}+n_{\mathsf{pub}}}$ is the vector consisting of i-th coordinates of all the public feature vectors, then $|\langle w, x_i \rangle| = y_i$.

In other words, if InstaHide only works with public feature vectors, then the problem of recovering which public vectors generated a given synthetic vector is formally *equivalent* to phase retrieval. In particular, if the public dataset is Gaussian, then we can leverage the existing algorithms for Gaussian phase retrieval. Huang et al. (2020) already noted this connection but argued that if InstaHide also uses private vectors, the existing algorithms for phase retrieval fail. Indeed, consider the extreme case where InstaHide only works with *private* feature vectors (i.e. $n_{\mathsf{pub}} = 0$), so that the only information we have access to is the synthetic vector $(y_1, ..., y_d)$ generated by InstaHide. In this case, it is clearly information-theoretically impossible to recover anything about w or the private dataset.

As we will see, the key workaround is to exploit the fact that InstaHide ultimately generates *multiple* synthetic vectors, each of which is defined by a *random* sparse convex combination of public/private vectors. And as we will make formal in Section 2.2, the right algorithmic question to study in this context can be thought of as a *multi-task*, *missing-data* version of phase retrieval (see Problem 2) that we believe to be of independent interest.

Lastly, we remark that in spite of this conceptual connection to phase retrieval, and apart from one component of our algorithm (see Section A.1) which draws upon existing techniques for phase retrieval, the most involved parts of our algorithm and its analysis utilize techniques that are quite different from the existing ones in the phase retrieval literature. We elaborate upon these techniques in Section 3.

1.2 Other Related Work

We refer to Huang et al. (2020) for an extensive discussion of how InstaHide connects to the broader distributed learning literature Shokri & Shmatikov (2015); McMahan et al. (2016); Konečnỳ et al. (2016); Kairouz et al. (2019) , data augmentation techniques like MixUp Zhang et al. (2018), and security notions like differential privacy Dwork et al. (2006; 2014) and instance-hiding schemes from cryptography Abadi et al. (1987). As mentioned above, our techniques are most closely connected to the Gaussian phase retrieval literature Cai et al. (2016), but we also remark that there are many works on multi-task learning in the context of regression Yi et al. (2014); Sedghi et al. (2016); Zhong et al. (2016); Li & Liang (2018); Chen et al. (2020), as well as on learning from data with missing entries Hardt (2014); Razenshteyn et al. (2016); Li et al. (2016), though these works do not pertain to phase retrieval and are not applicable to our setting.

2 Technical Preliminaries

Miscellaneous Notation Given a subset T, let $\mathcal{C}_T^k$ denote the set of all subsets of T of size exactly k. Given a vector $v \in \mathbb{R}^n$ and a subset $S \subseteq [n]$, let $[v]_S \in \mathbb{R}^{|S|}$ denote the restriction of v to the coordinates indexed by S.

Definition 2.1. Given a Gaussian distribution $\mathcal{N}(0, \Sigma)$, let $\mathcal{N}^{\mathsf{fold}}(0, \Sigma)$ denote the *folded Gaussian distribution* defined as follows: to sample from $\mathcal{N}^{\mathsf{fold}}(0, \Sigma)$, sample $g \sim \mathcal{N}(0, \Sigma)$ and output $|g|$.

2.1 The Generative Model

Definition 2.2 (Image matrix notation)**.** Let *image matrix* $\mathbf{X} \in \mathbb{R}^{d \times n}$ be a matrix whose columns consist of vectors $x_1, ..., x_n$ corresponding to n images each with d pixels taking values in $\mathbb{F}$.[3] It will also be convenient to refer to the rows of $\mathbf{X}$ as $p_1, ..., p_d \in \mathbb{R}^n$.

[3] We will often refer to public/private/synthetic feature vectors as *images*, and their coordinates as *pixels*, in keeping with the original applications of InstaHide to image datasets in Huang et al. (2020)

Definition 2.3 (Public/private notation)**.** Let $S \subset \{1, ..., n\}$ be some subset. We will refer to S and $S^c \triangleq \{1, ..., n\} \backslash S$ as the set of *public* and *private* images respectively, and given a vector $w \in \mathbb{R}^n$, we will refer to $\text{supp}(w) \cap S$ and $\text{supp}(w) \cap S^c$ as the *public* and *private coordinates* of w respectively.

Definition 2.4 (Synthetic images)**.** Given sparsity levels $k_{\mathsf{pub}} \leq |S|, k_{\mathsf{priv}} \leq |S^c|$, image matrix $\mathbf{X}$ and a *selection vector* $w \in \mathbb{R}^n$ for which $[w]_S$ and $[w]_{S^c}$ are k_{pub}- and k_{priv}-sparse respectively, the corresponding *synthetic image* is the vector

$$y^{\mathbf{X},w} \triangleq |\mathbf{X}w|, \tag{1}$$

where $|\cdot|$ denotes entrywise absolute value. We say that $\mathbf{X}$ and a sequence of selection vectors $w_1, ..., w_M \in \mathbb{R}^n$ give rise to a *synthetic dataset* consisting of the images $\{y^{\mathbf{X},w_1}, ..., y^{\mathbf{X},w_M}\}$.

Note that instead of the entrywise absolute value of $\mathbf{X}w$, InstaHide in Huang et al. (2020) randomly flips the sign of every entry of $\mathbf{X}w$, but these two operations are interchangeable in terms of information; it will be slightly more convenient to work with the former.

We will work with the following distributional assumption on the entries of $\mathbf{X}$:

Definition 2.5 (Gaussian images)**.** We say that $\mathbf{X}$ is a random *Gaussian image matrix* if its entries are sampled i.i.d. from $\mathcal{N}(0, 1)$.

We will also work with the following simple notion of "random convex combination" as our model for how the selection vectors $w_1, \ldots, w_M$ are generated:

Definition 2.6 (Distribution over selection vectors)**.** Let $\mathcal{D}$ be the distribution over selection vectors defined as follows. To sample once from $\mathcal{D}$, draw random subset $T_1 \subset S, T_2 \subseteq S^c$ of size k_{pub} and k_{priv} and output the unit vector whose i-th entry is $\frac{1}{\sqrt{k_{\mathsf{pub}}}}$ if $i \in T_1$, $\frac{1}{\sqrt{k_{\mathsf{priv}}}}$ if $i \in T_2$, and zero otherwise.[4]

The main algorithmic question we study is the following:

Problem 1 (Private image recovery). Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be a Gaussian image matrix. Given access to the public images $\{x_s\}_{s \in S}$ and the synthetic dataset $\{y^{\mathbf{X},w_1}, \ldots, y^{\mathbf{X},w_M}\}$, where $w_1, ..., w_M \sim \mathcal{D}$ are unknown selection vectors, output a vector $x \in \mathbb{R}^d$ for which there exists private image x_s (where $s \in S^c$) satisfying $|x_i| = |(x_s)_i|$ for all $i \in [d]$.

Remark 2.7. Note that it is information-theoretically impossible to guarantee that $x_i = (x_s)_i$. This is because the distribution over $\mathbf{X}$ and the distribution over matrices given by sampling $\mathbf{X}$ and multiplying every private image by -1 are both Gaussian. And if the selection vectors $w_1, ..., w_M$ generated the synthetic images in the former case, then the selection vectors $w'_1 \ldots, w'_M$, where w'_j is obtained by multiplying the private coordinates of w_j by -1, would generate the exact same synthetic images.

2.2 Multi-Task Phase Retrieval With Missing Data

In this section we make formal the discussion in Section 1.1 and situate it in the notation above. First consider a synthetic dataset consisting of a single image $y \triangleq y^{\mathbf{X},w}$, where w is arbitrary and $\mathbf{X}$ is a random Gaussian image. From Eq. (1) we know that

$$|\langle w, p_j \rangle| = y_j \ \forall \ j \in [d].$$

If $S = \{1, ..., n\}$, then the problem of recovering selection vector w from synthetic dataset $\{y\}$ is merely that of recovering w from pairs (p_j, y_j), and this is exactly the problem of *phase retrieval* over Gaussians. More precisely, because w is assumed to be sparse, this is the problem of *sparse phase retrieval* over Gaussians.

If $S \subsetneq \{1, ..., n\}$, then it's clearly impossible to recover the private coordinates of w from $y^{\mathbf{X},w}$ alone. But it may still be possible to recover the public coordinates: formally, we can

[4]Note that any such vector does not specify a convex combination, but this choice of normalization is just to make some of the analysis later on somewhat cleaner, and our results would still hold if we chose the vectors in the support of $\mathcal{D}$ to have entries summing to 1.

hope to recover $[w]_S$ given pairs $([p_j]_S, y_j)$, where the p_j's are sampled independently from $\mathcal{N}(0, \mathsf{Id}_n)$. This can be thought of as a *missing-data* version of sparse phase retrieval where some known subset of the coordinates of the inputs, those indexed by S^c, are unobserved.

But recall our ultimate goal is to say something about the *private images*. It turns out that because we actually observe multiple synthetic images, corresponding to multiple vectors w, it becomes possible to recover x_s for some $s \in S^c$ (even in the extreme case where $S = \emptyset$!). This corresponds to the following inverse problem which is *formally equivalent* to Problem 1, but phrased in a self-contained way which may be of independent interest.

Problem 2 (Multi-task phase retrieval with missing data). Let $S \subsetneq [n]$. Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be a matrix whose entries are i.i.d. draws from $\mathcal{N}(0,1)$, with rows denoted by $p_1, ..., p_d$ and columns denoted by $x_1, ..., x_n$. Let $w_1, \ldots, w_M \sim \mathcal{D}$.

For every $j \in [d]$, we get a tuple $\left([p_j]_S, y_j^{(1)}, \ldots, y_j^{(M)}\right)$ satisfying

$$|\langle w_i, p_j \rangle| = y_j^{(i)} \ \forall \ i \in [M], j \in [d].$$

Using just these, output $x \in \mathbb{R}^d$ such that for some $s \in S^c$, $|x_i| = |(x_s)_i|$ for all $i \in [d]$.

3 Proof Overview

At a high level, our algorithm has three components:

1. Learn the public coordinates of all the selection vectors $w_1, ..., w_M$ used to generate the synthetic dataset.
2. Recover the $M \times M$ rescaled Gram matrix $\mathbf{M}$ whose (i,j)-th entry is $k \cdot \langle w_i, w_j \rangle$.
3. Use $\mathbf{M}$ and the synthetic dataset to recover a private image.

Step 1 draws upon techniques in Gaussian phase retrieval, while Step 2 follows by leveraging the correspondence between the covariance matrix of a Gaussian and the covariance matrix of its corresponding *folded Gaussian* (see Definition 2.1). Step 3 is the trickiest part and calls for leveraging delicate properties of the distribution $\mathcal{D}$ over selection vectors.

Learning the Public Coordinates of Any Selection Vector We begin by describing how to carry out Step 1 above. First consider the case where $S = \{1, \ldots, n\}$, that is, where every image is public. Recall from the discussion in Section 1.1 and 2.2 that in this case, the question of recovering w from synthetic image $y^{\mathbf{X},w}$ is equivalent to Gaussian phase retrieval. One way to get a reasonable approximation to w is to consider the $n \times n$ matrix

$$\mathbf{N} \triangleq \mathop{\mathbf{E}}_{p,y}[y^2 \cdot (pp^\top - \mathsf{Id})], \qquad p \sim \mathcal{N}(0, \mathsf{Id}_n) \text{ and } y = |\langle w, p \rangle|.$$

It is a standard calculation (see Lemma A.3) to show that $\mathbf{N}$ is a rank-one matrix proportional to $ww^\top$. And as every one of $p_1, \ldots, p_d$ is an independent sample from $\mathcal{N}(0, \mathsf{Id})$, and $y_i^{\mathbf{X},w}$ satisfies $\langle w, p_i \rangle = y_i^{\mathbf{X},w}$ for every pixel $i \in [d]$, one can approximate $\mathbf{N}$ with the matrix

$$\widehat{\mathbf{N}} \triangleq \frac{1}{d} \sum_{i=1}^{d} \left(y_i^{\mathbf{X},w}\right)^2 \cdot (p_i p_i^\top - \mathsf{Id}).$$

This is the basis for the spectral initialization procedure that is present in many works on Gaussian phase retrieval, see e.g. Candes et al. (2015); Netrapalli et al. (2013). $\widehat{\mathbf{N}}$ will not be a sufficiently good spectral approximation to $\mathbf{N}$ when $d \ll n$, so instead we use a standard post-processing step based on the canonical SDP for sparse PCA (see (2)). Instead of taking the top eigenvector of $\widehat{\mathbf{N}}$, we can take the top eigenvector of the SDP solution and argue that as long as $d = \widetilde{\Omega}(\mathrm{poly}(k_{\mathsf{pub}}) \log n)$, this will be sufficiently close to w that we can exactly recover $\mathrm{supp}(w)$.

Now what happens when $S \subsetneq \{1, \ldots, n\}$? Interestingly, if one simply modifies the definition of $\mathbf{N}$ to be $\mathbf{E}_{p,y}[y^2 \cdot ([p]_S [p]_S^\top - \mathsf{Id})]$ and defines the corresponding empirical analogue $\widehat{\mathbf{N}}$

formed from the pairs $\{([p_i]_S, y_i^{\mathbf{X},w})\}_{i\in[d]}$, one can still argue (see Lemma A.3) that the $\mathbf{N}$ is a rank-1 $|S| \times |S|$ matrix proportional to $[w]_S[w]_S^\top$ and that the top eigenvector of the solution to a suitable SDP formed from $\widehat{\mathbf{N}}$ will be close to w (see Lemma A.4).

Recovering the Gram Matrix via Folded Gaussians As we noted earlier, it is information-theoretically impossible to recover $[w_i]_{S^c}$ for any $i \in [M]$ given only $y^{\mathbf{X},w_i}$ and $[w_i]_S$, but we now show it's possible to recover the inner products $\langle [w_i]_{S^c}, [w_j]_{S^c}\rangle$ for any $i, j \in [M]$. For the remainder of the overview, we will work in the extreme case where $S^c = \{1, ..., n\}$, though it is not hard (see Section A.7) to combine the algorithms we are about to discuss with the algorithm for recovering the public coordinates to handle the case of general S. For brevity, let $k \triangleq k_{\mathsf{priv}}$.

First note that the $M \times d$ matrix whose rows consist of $y^{\mathbf{X},w_1}, ..., y^{\mathbf{X},w_M}$ can be written as

$$\mathbf{Y} \triangleq \begin{pmatrix} |\langle p_1, w_1\rangle| & \cdots & |\langle p_d, w_1\rangle| \\ \vdots & \ddots & \vdots \\ |\langle p_1, w_m\rangle| & \cdots & |\langle p_d, w_m\rangle| \end{pmatrix}.$$

Observe that without absolute values, each column would be an independent draw from the M-variate Gaussian $\mathcal{N}(0, \mathbf{M})$, where $\mathbf{M}$ is the Gram matrix defined above. Instead, with the absolute values, each column of $\mathbf{Y}$ is actually an independent draw from the *folded Gaussian* $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{M})$ (Definition 2.1). The key point is that the covariance of $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{M})$ can be directly related to $\mathbf{M}$ (see Corollary A.6), so by estimating the covariance of the folded Gaussian $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{M})$ using the columns of $\mathbf{Y}$, we can obtain a good enough approximation $\widetilde{\mathbf{M}}$ to $\mathbf{M}$ that we can simply round every entry of $\widetilde{\mathbf{M}}$ so that the rounded matrix exactly equals $\mathbf{M}$. Furthermore, we only need to *entrywise* approximate the covariance of $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{M})$ for all this to work, which is why it suffices for d to grow *logarithmically* in M.

Discerning Structure From the Gram Matrix By this point, we have access to the Gram matrix $\mathbf{M}$. Equivalently, we now know for any $i, j \in [M]$ whether $\mathrm{supp}(w_i) \cap \mathrm{supp}(w_j) \neq \emptyset$, that is, for any pair of synthetic images, we can tell whether the set of private images generating one of them overlaps with the set generating the other. The crucial issue is that we have no way of discerning, e.g., whether any group of *three or more* synthetic images overlap in this way.

So for instance, even if we could find $k+1$ distinct indices $i_1, ..., i_{k+1}$ for which $|\mathrm{supp}(w_{i_j}) \cap \mathrm{supp}(w_{i_{j'}})| = k-1$, it is impossible even to distinguish between the case where the corresponding synthetic images all have the same $k-1$ private images in common, and the case where there is a group $T \subseteq [n]$ of $k+1$ private images such that each of these synthetic images is comprised of a subset of T of size k. This is the conundrum of typically not being able to recover the identity of a collection of subsets if one only knows the sizes of their pairwise intersections!

This leads to the following natural combinatorial question. What families of sets are *uniquely identified* (up to trivial ambiguities) by the sizes of the pairwise intersections? One answer to this question, as we show, is the family of all subsets of $\{1, ..., k+2\}$ of size k (see Lemma A.11). This leads us to the following definition:

Definition 3.1 (Floral Submatrices)**.** A $\binom{k+2}{k} \times \binom{k+2}{k}$ matrix $\mathbf{H}$ is *floral* if the following holds. Fix some lexicographic ordering on $\mathcal{C}^k_{[k+2]}$ and index $\mathbf{H}$ according to this ordering. There is some permutation matrix $\mathbf{\Pi}$ for which the matrix $\mathbf{H}' \triangleq \mathbf{\Pi}^\top \mathbf{H} \mathbf{\Pi}$ satisfies that for every pair of $S, S' \in \mathcal{C}^k_{[k+2]}$, $\mathbf{H}'_{S,S'} = |S \cap S'|$. See Example B.1 in the supplement.

The upshot is that if we can identify a floral submatrix of $\mathbf{M}$, then we know for certain that the subsets of private images picked by those selection vectors comprise all size-k subsets of some subset of $[n]$ of size $k+2$. In summary, using the pairwise intersection size information provided by the Gram matrix $\mathbf{M}$, we can pinpoint collections of selection vectors which share a nontrivial amount of common structure.

Learning a Private Image With a Floral Submatrix What can we do with this common structure in a floral submatrix? Given that the selection vectors $w_{i_1}, ..., w_{i_{\binom{k+2}{k}}}$ corresponding to the rows of the floral submatrix only involve $k+2$ different private images altogether, and there are $\binom{k+2}{k} > k+2$ constraints of the form $|\langle w_{i_j}, p_\ell \rangle| = y_\ell^{\mathbf{X}, w_{i_j}}$ for any pixel $\ell \in [d]$, we can hope that for each pixel, we can uniquely recover the $k+2$ private images from solving this system of equalities, where the unknowns are the values of the $k+2$ private images at that particular pixel. *A priori*, the fact that the number of constraints in this system exceeds the number of unknowns does not immediately guarantee that this system has a unique solution up to multiplying the solution uniformly by -1. Here we exploit the fact that $\mathbf{X}$ is Gaussian to show however that this is the case almost surely. Finally, note that this system can be solved in time $\exp(O(k^2))$ by simply enumerating over $2^{\binom{k+2}{k}}$ sign patterns. We conclude that if we could find a floral submatrix, then we would find not just one, but in fact $k+2$ private images!

A Floral Submatrix Exists, and How to Find It It remains to understand how big M has to be before we can guarantee the existence of a floral submatrix inside the Gram matrix $\mathbf{M}$ with high probability. Obviously if M were big enough that with high probability we see *every possible* synthetic image that could arise from a selection vector w in the support of $\mathcal{D}$, then $\mathbf{M}$ will contain many floral submatrices. One surprising part of our result is that we can ensure the existence of a floral submatrix when M is much smaller. Our proof of this is quite technical, but at a high level it is based on the second moment method (see Lemma A.12).

The final question is: provided a floral submatrix of $\mathbf{M}$ exists, *how do we find it*? Note that naively, we could always brute-force over all $\binom{M}{O(k^2)} \leq n^{O(k^3)}$ principal submatrices with exactly $\binom{k+2}{k}$ rows/columns, and for each such principal submatrix we can check in $\exp(\widetilde{O}(k))$ time whether it is floral.

Surprisingly, we give an algorithm that can identify a floral submatrix of $\mathbf{M}$ in time dominated by the time it takes to write down the entries of the Gram matrix $\mathbf{M}$. Note that an off-the-shelf algorithm for subgraph isomorphism would not suffice as the size of the submatrix in question is $O(k^2)$, and furthermore such an algorithm would need to work for *weighted graphs*. Instead, our approach is to use the constructive nature of the proof in Lemma A.11, that the family of all subsets of $\{1, ..., k+2\}$ of size k is uniquely identified by the sizes of the pairwise intersections. By algorithmizing this proof, we give an efficient procedure for finding a floral submatrix, see Algorithm 3 and Lemma A.17. An important fact we use is that if we restrict our attention to the entries of $\mathbf{M}$ equal to $k-1$ or $k-2$, this corresponds to a graph over the selection vectors which is sparse with high probability.

We defer the formal specification and analysis of our algorithm to the supplement.

4 Experiments

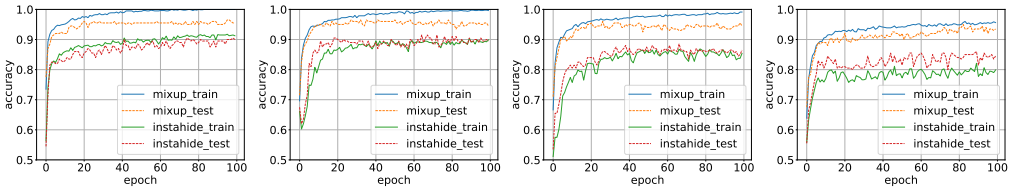
We describe an experiment demonstrating the *utility* of InstaHide for Gaussian images and comparing to the utility of another data augmentation scheme, MixUp Zhang et al. (2018). We also informally report on our implementation of LearnPublic and its empirical efficacy.

4.1 Choice of Architecture and Parameters

As our empirical results are purely for proof-of-concept, we work with a fairly basic neural network architecture. We use a 4-layer neural network as a binary classifier,

$$y = \arg\max(\text{softmax}(W_4 \sigma(W_3 \sigma(W_2(\sigma(W_1 x + b_1)) + b2) + b3) + b4)),$$

where $x \in \mathbb{R}^{10}$, $W_1 \in \mathbb{R}^{100\times 10}$, $W_2 \in \mathbb{R}^{100\times 100}$, $W_3 \in \mathbb{R}^{100\times 100}$, $W_4 \in \mathbb{R}^{2\times 100}$, $b_1 \in \mathbb{R}^{100}$, $b_2 \in \mathbb{R}^{100}$, $b_3 \in \mathbb{R}^{100}$, $b_4 \in \mathbb{R}^{100}$. We model $W_l \sim \mathcal{N}(u_l, 1)$, $b_l \sim \mathcal{N}(u_l, 1)$, $u_l \sim \mathcal{N}(0, \alpha)$. We train the neural network for 100 epochs with cross-entropy loss and SGD optimizer with a learning rate of 0.01. We do not need to distinguish between public and



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 6$

Figure 1: Comparing MixUp and Instahide training on Gaussian dataset with different k.

private images in our experiments, so let $k_{\mathsf{priv}} = 0$ and $k_{\mathsf{pub}} = k$ for $k \in \{1, 2, 3, 6\}$, and for each choice of k, we use random k-sparse selection vectors whose nonzero entries equal $1/k$. In all of our experiments, we separate our original image data (before generating synthetic data) into two categories: 80% training data and 20% test data. We train on synthetic images generated by MixUp or InstaHide using the training data, and measure the "test accuracy" on the training data and the test data separately. We provide more choices of k in Appendix C.

4.2 Gaussian Data

Settings We considered binary classification on Gaussian data. We generated random images $x_1, \ldots, x_{1000} \in \mathbb{R}^{10}$ from $\mathcal{N}(0, \mathsf{Id})$ and a random vector $v \in \mathbb{R}^{10} \sim \mathcal{N}(0, \mathsf{Id})$. We then ranked all the Gaussian images based on $\sum_i |x_i v_i|$ and labeled the largest half as '1' and the rest as '0'. The point of choosing this labeling function is that it would assign the same label to any x, x' which agree entrywise in magnitudes. Given a synthetic image generated via MixUp or InstaHide using selection vector w, we assigned it the label which is the convex combination of the one-hot encodings of the labels of the original images indexed by w.

Results We compare training and test loss over epochs when training on a synthetic dataset generated by either MixUp or Instahide, as shown in Figure 1. We use the convention in this paper of defining synthetic images under InstaHide to have all nonnegative entries (rather than imposing random sign flips), though we explore in the supplement how random sign flips can affect learnability. Compared to training on MixUp, InstaHide results on lower model performance (accuracy). As we expected, when we increase the k, both MixUp and InstaHide suffer from accuracy loss. Instahide dropped by $\sim 10\%$ accuracy when $k = 6$ compared to classical training $k = 1$, while MixUp dropped by $\sim 5\%$.

4.3 Implementation of LearnPublic

We implemented LearnPublic for $k_{\mathsf{priv}} = 2$ and $n \in \{2000, 5000, 7500, 10000\}$. For $k_{\mathsf{pub}} = 2, 4, 6$, we respectively chose $d = 1000, 1800, 2400$. In particular, our choice of d is meant to work essentially for any choice of n (modulo the logarithmic dependence which does not noticeably manifest in this regime). One heuristic modification that we made to LearnPublic was to use a *diagonal thresholding* approach from Cai et al. (2016) in place of solving the SDP in (2): namely for every $j \in [n]$ we computed the quantity $\frac{1}{d}\sum_{i=1}^{d} y_i^2 \cdot (x_i)_j^2$, zeroed out all but the principal submatrix of $\widetilde{\mathbf{M}}$ indexed by the top 25 such j, and computed the top eigenvector of the resulting matrix. For each parameter setting we found that, as expected, we were able to recover an average of at least 90% of the support. As this experiment was primarily to demonstrate that d can be much less than n, we did not explore further optimizations to the algorithm.

References

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

T Tony Cai, Xiaodong Li, and Zongming Ma. Optimal rates of convergence for noisy sparse phase retrieval via thresholded wirtinger flow. *The Annals of Statistics*, 44(5):2221–2251, 2016.

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

Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

Sitan Chen, Jerry Li, and Zhao Song. Learning mixtures of linear regressions in subexponential time via fourier moments. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 587–600, 2020.

Aldo Conca, Dan Edidin, Milena Hering, and Cynthia Vinzant. An algebraic characterization of injectivity in phase retrieval. *Applied and Computational Harmonic Analysis*, 38 (2):346–356, 2015.

Ilias Diakonikolas, Daniel M Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and sq lower bounds for pac learning one-hidden-layer relu networks. In *Conference on Learning Theory (COLT)*, pp. 1514–1539, 2020.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 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.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. 2014.

Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. In *ICLR*. arXiv preprint arXiv:1711.00501, 2018.

Moritz Hardt. Understanding alternating minimization for matrix completion. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 651–660. IEEE, 2014.

Moritz Hardt and Eric Price. Tight bounds for learning a mixture of two gaussians. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pp. 753–760, 2015.

Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *ICML*, 2020.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Raymond Kan and Cesare Robotti. On moments of folded and truncated multivariate normal distributions. *Journal of Computational and Graphical Statistics*, 26(4):930–934, 2017.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

Yuanzhi Li and Yingyu Liang. Learning mixtures of linear regressions with nearly optimal complexity. In *COLT*, 2018.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with ReLU activation. In *Advances in neural information processing systems (NIPS)*, pp. 597–607, 2017.

Yuanzhi Li, Yingyu Liang, and Andrej Risteski. Recovery guarantee of weighted low-rank approximation via alternating minimization. In *International Conference on Machine Learning (ICML)*, pp. 2358–2367, 2016.

H Brendan McMahan, Eider Moore, Daniel Ramage, and Seth Hampson. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 93–102. IEEE, 2010.

Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. Phase retrieval using alternating minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2796–2804, 2013.

Matey Neykov, Zhaoran Wang, and Han Liu. Agnostic estimation for misspecified phase retrieval models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4089–4097, 2016.

Ilya Razenshteyn, Zhao Song, and David P Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th Annual Symposium on the Theory of Computing (STOC)*, 2016.

Hanie Sedghi, Majid Janzamin, and Anima Anandkumar. Provable tensor methods for learning mixtures of generalized linear models. In *Artificial Intelligence and Statistics*, pp. 1223–1231, 2016.

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

Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating minimization for mixed linear regression. In *International Conference on Machine Learning (ICML)*, pp. 613–621, 2014.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Kai Zhong, Prateek Jain, and Inderjit S Dhillon. Mixed linear regression with multiple components. In *Advances in neural information processing systems*, pp. 2190–2198, 2016.

Kai Zhong, Zhao Song, and Inderjit S Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. In *arXiv preprint*, 2017a.

Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, 2017b.

Kai Zhong, Zhao Song, Prateek Jain, and Inderjit S Dhillon. Provable non-linear inductive matrix completion. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11439–11449, 2019.