DAUnTLeSS: Data Augmentation and Uniform Transformation for Learning with Scalability and Security

Hanshen Xiao and Srinivas Devadas

MIT, {hsxiao,devadas}@mit.edu

Abstract. We revisit private optimization and learning from an information processing view. The main contribution of this paper is twofold. First, different from the classic cryptographic framework of operation-by-operation obfuscation, a novel private learning and inference framework via either data-dependent or random transformation on the sample domain is proposed. Second, we propose a novel security analysis framework, termed probably approximately correct (PAC) inference resistance, which bridges the information loss in data processing and prior knowledge. Through data mixing, we develop an information theoretical security amplifier with a foundation of PAC security.

We study the applications of such a framework from generalized linear regression models to modern learning techniques, such as deep learning. On the information theoretical privacy side, we compare three privacy interpretations: ambiguity, statistical indistinguishability (Differential Privacy) and PAC inference resistance, and precisely describe the information leakage of our framework. We show the advantages of this new random transform approach with respect to underlying privacy guarantees, computational efficiency and utility for fully connected neural networks.

1 Introduction

Machine Learning (ML) is a generic concept to build mathematical models based on data (samples) to make predictions or decisions. Within the broad category of data analysis, private ML is receiving significant attention. Depending on whether analysis results (outputs) need to be disclosed or not, there are two typical secure learning scenarios: One can be modelled as that a user asks a server to securely compute some function of its data without information leakage on either the input (data) or output (result). The other has a different goal, which is to disclose the result of a function on a given data set but trading off the dataset privacy and result utility, for example, the Census Bureau publishes the average salary of the population in the Boston area.

As for the former case, homomorphic encryption (HE) provides a solution. HE allows computation over encrypted data and only the user with the secret key can decrypt the output [AAUC18, MSM17]. Many existing secure learning works follow a protocol where optimization is implemented over encrypted data (ciphertext). Theoretically, if an honest server follows the protocol, the final decrypted model/output matches the result of the training as if it had been performed on the plaintext data.

As for the second case, where results are disclosed, Differential Privacy (DP) [DMNS06] is a representative example. Most existing DP frameworks utilize an additive noise to randomize an algorithm such that from its outputs, it is hard for one to distinguish the participation of a single datapoint. The noise added should be in a scale of the *sensitivity* of an algorithm. In the DP context, sensitivity is defined as the worst-case difference on the output of the algorithm that two adjacent datasets can produce. Therefore, through the noise, one can control the privacy loss. Existing DP setups generally require computation to be implemented by the data holder so only aggregate information is exposed: for example, gradient descent perturbed with well-scaled noise across iterations [BST14] can be used to train a private model. In general, determining the sensitivity

of an algorithm is an NP-hard problem [XT08]. Even with the assistance of homomorphic encryption, if a neural network is securely trained via a cloud service, making the model differentially private when applying it for inference is not easy.

If a user just wishes to privately release her data rather than privately computing, the only formal privacy notion as far as we know is Local DP (LDP) [DJW13], rooted in randomized response [War65]. In the context of LDP, each attribute of the user's data has to be perturbed before release, which makes distinguishing *every* single sample hard. LDP comes with a tight and heavy cost that the perturbation added to each sample is in a scale of $\Theta(d)$, where d denotes the dimensionality. Provided such heavily noisy data, most medium-size learning tasks become impossible [PCS$^+$20]. Therefore, this raises an interesting question: how can a user first privately release her data and resort to outside computation power for model training, while afterwards, the trained model can be also privately used for further prediction or decision? One straightforward approach could be a hybrid of cryptography and DP, where the model is first trained over homomorphically encrypted data and then the model is perturbed every time it is used. This apparently universal framework indeed encounters some obstacles in practice. Besides the above-mentioned sensitivity challenge, so far, even partial homomorphic encryption based learning protocols encounter large computational and communication overhead. Even training over MNIST data will take tens of hours in many state-of-art works [WGC19, ZLX$^+$20] with dozens of gigabytes of traffic to have reasonable accuracy. From an information processing perspective, in this paper, we set out to propose a unified framework to preserve security in both scenarios described earlier.

In the context of private learning, one key fact that sometimes gets overlooked is that the purpose of learning is to develop efficiently a *usable* model, where either the optimum of the model or the approaches to construct the model may not be *unique*. As mentioned earlier, through (partial) homomorphic encryption, one can securely obtain the exact same model as when the training algorithm is applied on the original data. Differently, the framework proposed in this paper implements the training procedure on a transformed sample domain to produce a usable model.

We start from a view of statistical supervised learning where we assume samples $s_i = (x_i, y_i)$, $i = 1, 2, ..., n$, are i.i.d. from certain distribution P, which we lack full knowledge of in most cases. Here, $x_i \in \mathcal{X}$ denotes the feature and $y_i \in \mathcal{Y}$ corresponds to the label. The goal of training is to find a model f^* such that

$$f^* = \arg\min_f \mathbb{E}_{(x,y)\sim P}\, l(f(x), y). \tag{1}$$

Here, $l(\cdot, \cdot)$ is some loss function selected, for example, square loss, where $l(f(x), y) = \|f(x) - y\|^2$. Given the model f, for newly incoming feature x_+, one can make a prediction or decision based on $f(x_+)$ to approximate the true label y_+. For a certain model $f(\cdot)$, usually, we call $\mathbb{E}_{(x,y)\sim P}\, l(f(x), y)$ as the generalization error of $f(\cdot)$ and $\frac{1}{n}\sum_{i=1}^n l(f(x_i), y_i)$ as the empirical error.

Taking the generalization error as the performance metric of a model, the following observation shows the relationship amongst a class of optimal models in different sample domains. Consider applying a transform T on the sample (feature) domain, where instead our focus becomes figuring out f_T^* such that

$$f_T^* = \arg\min_f \mathbb{E}_{(x,y)\sim P}\, l(f(T(x)), y). \tag{2}$$

Given the relationship between f^* and f_T^*, we have the following fact: *If the transformation T is bijective and the optimum of (1) is unique, then $f_T^* = f^* \circ T^{-1}$.* We then have the following identity that

$$\mathbb{E}_{(x,y)\sim P}\, l(f(T(x)), y) = \mathbb{E}_{(x,y)\sim P}\, l(f \circ T^{-1}((T(x)), y).$$

Therefore, theoretically, under an arbitrary bijection transform over $\mathcal{X}$, an equivalent transformed optimal model exists. This provides a natural secure learning framework, where the transformation T can be viewed

as a key and $T(x)$ acts as an encryption procedure. Ideally, one can select some one-way function and the information leakage on $T(x)$ is negligible at least in a sense of computational hardness.

A workflow of proposed framework is presented in Fig. 1, which is similar to homomorphic encryption but in an information theoretical way. On the user side, after some preprocessing (such as normalization or data augmentation) [1], data is *uniformly* transformed by a function $T(\cdot)$; On the server side, a model $f(\cdot)$ is trained over the transformed data which will then be sent back to the user; Finally, the user takes $f \circ T(\cdot)$ as the reconstructed model for further prediction or decision. However, this apparently simple and elegant

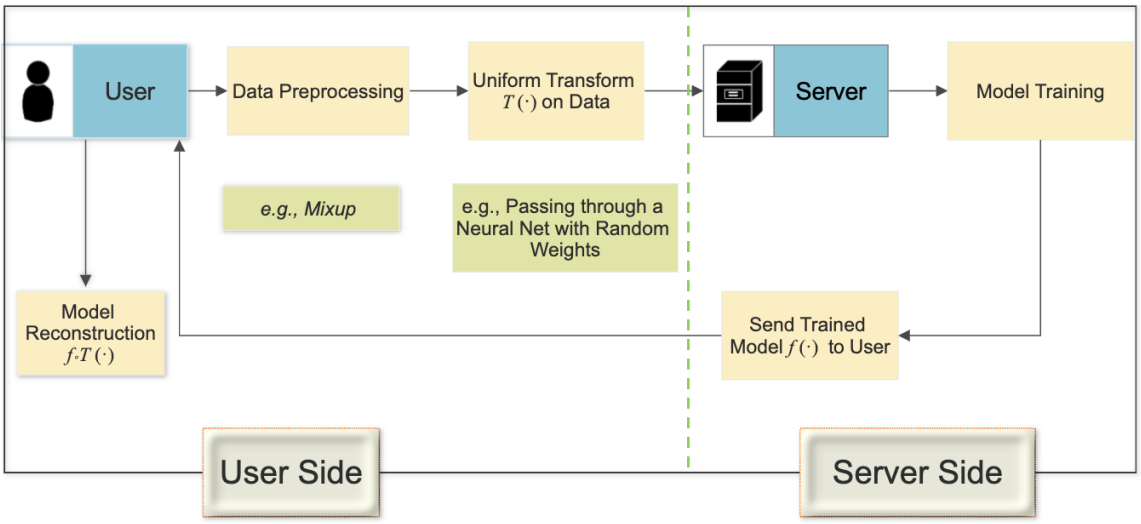


Fig. 1: The Dauntless Framework for Private Learning

framework faces three challenges.

1.1 Challenges

Proper Transformation: First, the above framework relies on an assumption that a magically powerful learning protocol exists which can always discover the optimal model in the stochastic optimization of (1). However, in practice, there is limited or even no prior knowledge regarding the sample distribution P, which instead needs to be approximated by samples observed. In an extreme case, one can imagine, without any prior knowledge on P, after applying a (pseudo)-random function ((P)RF) on $\mathcal{X}$, the distribution of transformed samples $T(P)$ may not be concentrated. Transformed samples $T(x)$ only carry the information of $T(P)$ at selected discrete sampled points while the generalization to un-sampled points becomes impossible. [2] A second related issue is the learning capacity. In practice, optimization of (1) is over a bounded class of models f rather than arbitrary functions. The class of candidate models is characterized with finite parameters. For

[1] Temporally, we may take the sample $s = (x, y)$ defined above as data after preprocessing for simplicity.

[2] From a view of sampling theory, without proper prior knowledge, finite samples can never be a sufficient interpretation of an arbitrary population.

example, if the generation of (x, y) can be captured by a noisy linear model where $y = \langle a, x\rangle + e$ for some noise e, and we set out to find the optimal linear regression model, then (1) becomes

$$\omega^* = \arg\min_{\omega} \mathbb{E}_{(x,y)\sim P}\, l(\langle \omega, x\rangle, y). \tag{3}$$

Clearly, if one applies some quadratic function T on x, then $T(x)$ and y may not be well approximated by any linear model. In other words, after transformation, a linear regression model may not be suitable any more and we need a larger function class to guarantee controllable generalization error. Therefore, a proper transformation T should satisfy that, with high confidence, transformed model $f^* \circ T^{-1}$ can still be well approximated within the class of candidate functions C_f, where C_f is dependent on the training capacity and strategy of a learner.

Multiple Encryptions with the Same Key: It is well known that if one applies the same key to encrypt multiple messages, the ciphertext has the potential to be statistically dependent and thus may be vulnerable to be attacked. Since the transformation T has to be carried out uniformly over the sample domain, a composition of privacy loss across all transformed samples $T(x)$ is unavoidable. In large-scale learning, public data may be utilized for training, where the adversary can conduct a "chosen-plaintext attack".

Privacy v.s. Utility Dilemma or Trapdoor: Information theoretical privacy is a mature field, where many statistical metrics have been developed to quantify the hardness of distinguishing the true input. However, indistinguishability never comes for free. Many (tight) lower bounds between utility and the indistinguishability measurement are known, for example in convex optimization [BST14] and Principal Component Analysis (PCA) [CSS12], [DTTZ14]. Notably, the accompanying utility compromise when naively releasing data with reasonable LDP is often unaffordable.

Similarly, in the transformation framework proposed, if T renders stronger privacy preservation, intuitively, the preimage of $T(x)$ should be harder to recover or distinguish, which simultaneously also weakens the learning efficiency over transformed data $T(\mathcal{X})$. Although uniform transformation seems like a smarter way to obfuscate data compared to purely random noise, so far it is still a strict tradeoff between utility and privacy. Ideally, it is expected that such transformation T behaves like a trapdoor where the user who holds T enjoys efficient learning while the adversary without T must *pay much more* compared to the user's cost. We know in cryptography, NP complete problems can be built as trapdoors in an encryption scheme, where *what the adversary pays* is quantified as the computational complexity. In the context of information processing without restrictions on computation power, will a trapdoor still be possible?

1.2 Paper Organization and Contributions

We set out to address the three challenges posed above. In general, there are two possible kinds of transformations to meet the requirements. One kind of transformation can be data-dependent which maps the private data to some fixed or independent output, for example, some public data. In this case, the transformed data is totally independent of the private data, and perfect secrecy is achievable. However, it is noted that no information is propagated through the transformation and such operation usually needs some specific assumptions, like some proper prior knowledge on the sample distribution P. In Section 2, we will provide a concrete example for linearly separable data.

The second can be modelled as a random selection of T from a set of functions C_T of good continuity and locality. This allows information carried by data to be propagated through the transmission channel. We focus on this approach in Section 3, where we provide a protocol for data mixing and random transformation that uses the entropy of private data as an information-theoretic security amplification. In Section 3, we first describe the three privacy interpretations we will use to analyze the protocol, namely, ambiguity, statistical

indistinguishability and Probably Approximately Correct (PAC) inference resistance, and then describe the protocol. In Section 4, we analyze our protocol and provide security guarantees under these three privacy interpretations.

In Section 5, we provide results obtained using transformed data training as compared to non-private data training. We show perfect secrecy can be achieved for learning over linearly separable data, and show that transformed data learning achieves computational efficiency and utility for fully connected networks. In Section 6, we present a unified framework to accommodate both private learning and private inference, controlling both data privacy leakage as well as the privacy of the user's model.

In Section 7, we review related works, and conclude in Section 8.

2 Warm-up: Generalized Linear Model

As a warm-up, we start with a binary classifier with the assumption that the feature space is linearly (hyperplane) separable. To be formal, we say two sets $\mathcal{X}_1$ and $\mathcal{X}_2$ whose elements are in $\mathbb{R}^d$ are linearly separable if there exist $\omega^*, b^* \in \mathbb{R}^{1\times d}$ such that $x \cdot (\omega^*)^T > b^*$ for $x \in \mathcal{X}_1$ while $x \cdot (\omega^*)^T < b^*$ for $x \in \mathcal{X}_2$. So, it is natural to ask what kind of transformation T can preserve the linear separability of $T(\mathcal{X}_1)$ and $T(\mathcal{X}_2)$. The following lemma provides a sufficient condition:

Lemma 1. *If the transformation $T(\cdot)$ is an injective affine mapping, then for $\mathcal{X}_1, \mathcal{X}_2 \in \mathbb{R}^d$ which are linearly separable, $T(\cdot)$ preserves linear separability.*

Proof. If an affine transformation represented by $T(x) = xA + c$ is injective, it is equivalent to that AA^T should be of full rank. We assume that a line $x\cdot(\omega^*)^T - b^*$ separates the two sets $\mathcal{X}_1$ and $\mathcal{X}_2$, i.e., $x\cdot(\omega^*)^T > b^*$ for $x \in \mathcal{X}_1$ while $x \cdot (\omega^*)^T < b^*$ for $x \in \mathcal{X}_2$. Now, consider the following identity:

$$x \cdot (\omega^*)^T = (xA) \cdot A^T(AA^T)^{-1}(\omega^*)^T.$$

Thus, in the codomain of $T(\cdot)$, $\langle T(x), A^T(AA^T)^{-1}(\omega^*)^T\rangle - c\cdot A^T(AA^T)^{-1}(\omega^*)^T - b$ still separates $T(\mathcal{X}_1)$ and $T(\mathcal{X}_2)$.

As a special case, if we set T to be a bijective linear transformation in a form $T(x) = xA$ for some invertible matrix $A \in \mathbb{R}^{d\times d}$, then it is not hard to verify the following: if $\mathcal{X}_1$ and $\mathcal{X}_2$ can be separated by a hyperplane $x \cdot (\omega^*)^T = b^*$, then the hyperplane $x \cdot A^{-1}(\omega^*)^T = b^*$ separates $T(\mathcal{X}_1)$ and $T(\mathcal{X}_2)$. For data which can be almost linearly separable under small noise perturbation, a classifier can be efficiently found through a linear regression. Indeed, the above separability preservation intuition can be further generalized, where the data generation can be described in a form

$$y = g(\langle \omega^*, x\rangle + b^*) \tag{4}$$

for some function $g(\cdot)$ with parameters ω^* and b^*. (4) is called Generalized Linear Model (GLM). Interested readers may refer to [NW72] which elaborates the various applications of GLM. Under the prior knowledge that data can be well captured by some GLM, a standard approach to approximate (ω^*, b^*) is through Empirical Risk Minimization (ERM), i.e.,

$$(\omega_e^*, b_e^*) = \arg\min_{\omega,b} \sum_{i=1}^{n} l(g(\langle \omega, x_i\rangle + b), y_i). \tag{5}$$

The key observation w.r.t. GLM is the following identity: for an arbitrary matrix A such that its right inverse exists, i.e., $AA_r^{-1} = \boldsymbol{I}_d$, then $\langle xA, A_r^{-1}\omega^T\rangle = \langle \omega, x\rangle$. Therefore, with a similar reasoning, if the data

generation can be well approximated by some GLM, after an injective linear transform on the sample domain, the true optimum is recoverable from the optimization over transformed data.

To formalize the analysis, consider applying a right-invertible matrix A on each sample x_i and construct the ERM problem on transformed samples as

$$(\omega_{A,e}^*, b_{A,e}^*) = \arg\min_{\omega,b} \sum_{i=1}^{n} l(g(\langle \omega, x_i A \rangle), y_i). \tag{6}$$

Clearly, if the optimum (ω_e^*, b_e^*) of (5) is unique, then $(\omega_{A,e}^*)^T = A_r^{-1}(\omega_e^*)^T$ while $b_{A,e}^* = b_e^*$. Please note that this does not require any assumptions on the function $l(\cdot)$ and $g(\cdot)$ such as convexity or smoothness [3].

2.1 Perfect Secrecy

It should now be clear that proper transformation can still guarantee the learnability of a class of transformed data. But what kind of privacy preservation can be provided by a linear transformation? In this section, we stick to an intuitive perfect privacy preservation. We say a (randomized) mechanism has perfect secrecy or security if the output (distribution) is fully independent of its input.

Definition 1. *A mechanism $\mathcal{M}$ is perfectly secure if for arbitrary input I, the distribution of output $\mathcal{M}(I)$ is independent of I.*

If so, observation on the output $\mathcal{M}(I)$ cannot provide any further information with respect to (w.r.t.) the sensitive input I. In the following, we will show that efficient learning with perfect privacy is achievable in linearly-separable data. For the input matrix $X \in \mathbb{R}^{n\times d}$ which contains n samples each within $\mathbb{R}^d$, we first consider a special case where $n = d$ and X is invertible. It is noted that once X^{-1} exists, the transformation $T_{X^{-1}}(x) = xX^{-1}$ is a bijective mapping over $\mathbb{R}^d$. On the other hand, $T_{X^{-1}}(X)$ always returns the identity $\boldsymbol{I}_d$, a constant, which achieves prefect privacy w.r.t. input X. In practice, the number of samples n is usually larger than the dimensionality d. To generalize the above idea, for input matrix $X \in \mathbb{R}^{n\times d}$ with $n \geq d$, we consider the following transformation $\hat{X} = X \cdot A$, for some matrix $A \in \mathbb{R}^{d\times n}$ of rank d to expand X to be a square matrix. Clearly, $\hat{X}$ is singular (of rank at most d) and we cannot find the inverse of $\hat{X}$. However, it can be proved that after an arbitrarily small perturbation on $\hat{X}$, we can make it invertible.

Lemma 2. *For a singular matrix $\hat{X} \in \mathbb{R}^{n\times n}$, $\hat{X} \neq \mathbf{0}$, of which the least non-zero eigenvalue in absolute value is γ_0, then for any γ such that $|\gamma| < \gamma_0$, $\hat{X} + \gamma \boldsymbol{I}_n$ is invertible.*

Proof. We use the following identity that

$$det(\gamma \cdot \boldsymbol{I} - \hat{X}) = \prod_{i=1}^{n} (\gamma - \gamma_i),$$

[3] In linear regression for heavy-tailed distributions, usually the loss function has to be selected to be non-convex, for example, in [ZZ18], the loss function is defined as $\frac{1}{\alpha}\psi(\alpha|y_i - x_i \cdot \omega^T|)$, where

$$-\log(1 - z + z^2/2) \leq \psi(z) \leq \log(1 + z + z^2/2)$$

for some positive number α. This complicated min-max estimator is shown to be more robust but also requires intensive computation [AC11].

where $\gamma_{[1:n]}$ denote the eigenvalues of $\hat{X}$ in an ascending order. Clearly, if $\gamma \cdot \boldsymbol{I} - \hat{X}$ is invertible, then $det(\gamma \cdot \boldsymbol{I} - \hat{X}) \neq 0$. On the other hand, to guarantee $det(\gamma \cdot \boldsymbol{I} - \hat{X}) \neq 0$, a sufficient condition is $\gamma \neq \gamma_i$ for any i and $\gamma \neq 0$. For n discrete reals $\gamma_{[1:n]}$, one can always find an arbitrarily small but nonzero γ such that it is not equal to any eigenvalue of $\hat{X}$.

To summarize, the above described a perfectly secure protocol from $\mathbb{R}^{n\times d} \to \boldsymbol{I}_n$ in three steps. First, one randomly selects $A \in \mathbb{R}^{d\times n}$ and computes $\hat{X} = XA$. Second, a small perturbation is added to $\hat{X}$ to form $\tilde{X} = \hat{X} + \gamma \boldsymbol{I}_n$. Third, take the inverse of $\tilde{X}$, i.e., $\tilde{X}^{-1}$, as the multiplicative transform, which maps the data to the constant $\boldsymbol{I}_n$. Here, we assume that the labels are constant. For example, in a binary classification problem, one always randomly encodes the two classes by 0 and 1 and the ratio between the numbers of respective samples is set to be 1:1. Thus, the exposed information is independent of the input and perfect secrecy is achieved.

2.2 Utility Analysis and Regularization

We say two sets $\mathcal{X}_1, \mathcal{X}_2 \in \mathbb{R}^d$ are δ-strongly linearly separable, if, besides the existence of a hyperplane separating them, their margin satisfies $\min_{x_1\in\mathcal{X}_1, x_2\in\mathcal{X}_2} \|x_1 - x_2\| \geq \delta > 0$. To obtain a straightforward corollary of Lemma 2, when $\mathcal{X}_1$ and $\mathcal{X}_2$ are δ-strongly linearly separable, consider the above mentioned transformation $XA + \gamma \boldsymbol{I}_n$ on input dataset X. Assuming $A \in \mathbb{R}^{d\times n}$ of rank d where the least singular value of A is σ_0, then for transformed $\mathcal{X}_1$ and $\mathcal{X}_2$,

$$\min_{x_1\in\mathcal{X}_1, x_2\in\mathcal{X}_2} \|(x_1 - x_2)A\| \geq \sigma_0\delta.$$

Therefore, after a $\gamma\boldsymbol{I}$ perturbation on X, the l_2 norm of perturbation on each sample is at most γ and thus the minimal distance of transformed $\mathcal{X}_1$ and $\mathcal{X}_2$ under perturbation is at least $\sigma_0\delta - 2\gamma$.

With a similar reasoning, for GLM optimization, let $GL(\omega, b) = \sum_{i=1}^n l(g(\langle\omega, x_i\rangle + b), y_i)$ while $GL_{A,\gamma}(\omega, b) = \sum_{i=1}^n l(g(\langle A_r^{-1}\omega^T, x_iA + \gamma e_i\rangle + b), y_i)$, where e_i is the i^{th} natural basis of $\mathbb{R}^n$. Thus, with some mild Lipschitz assumptions on functions $l(\cdot)$ and $g(\cdot)$, the difference $GL(\omega, b) - GL_{A,\gamma}(\omega, b)$ can be well controlled while the perturbation γ can be arbitrarily small. The abovementioned techniques can also be applied in multinomial regression for multi-class problems; we explore this in Section 5.1.

Usually, to avoid overfitting, instead of solving ERM directly, we will add a regularization term, say $\lambda\cdot\|\omega\|$, at the end of (5) with some positive constant λ. However, it is noted that, after a multiplicative transformation, straightforwardly adding a regularization term in (6) may not help. If the inverse of transformed data $\tilde{X}^{-1}$ is close to being singular where the largest singular value of $\tilde{X}^{-1}$ is huge, then it is hard to control the normal regularization term when handling the optimization over the transformed sample domain. Therefore, a better way is to mimic Principal Component Analysis (PCA) to improve the small singular values of A. But different from PCA, we do not remove the small singular values which results in dimensionality reduction; instead, we increase the small ones to a certain threshold. The following method in Algorithm 1 is called Singular Value Improvement (SVI).

The parameter τ behaves like the regularization coefficient: a larger τ implies weaker singularity. Moreover, it is noted that UV, a product of two unitary matrices is still unitary. Thus, SVI with parameter τ introduces at most a perturbation per row whose l_2 norm is upper bounded by τ. A similar utility analysis can be derived.

2.3 Deep Learning with Perfect Secrecy

In general, a neural network can be described as a function of multi-layer GLM:

$$\mathcal{N}_{\boldsymbol{W}}(x) = \sigma(..., \sigma(\sigma(x \cdot W_1) \cdot W_2), ..., W_L) \tag{7}$$

Algorithm 1 Singular Value Improvement (SVI)

Input: Matrix $A \in \mathbb{R}^{l \times l'}$; Threshold parameter $\tau > 0$.

1: Apply SVD on $A = U\Sigma V$, where Σ is the singular value diagonal matrix;
2: Change all diagonal elements in Σ smaller than τ to τ and obtain $\bar{S}$.
3: Construct $\bar{A} = U\bar{\Sigma}V$.

Output: Matrix $\bar{A} \in \mathbb{R}^{l \times l'}$.

where σ is the selected activation function, say ReLu or Sigmoid functions, and $\boldsymbol{W} = (W_1, W_2, ..., W_L)$, $W_1 \in \mathbb{R}^{d \times d_1}, W_2 \in \mathbb{R}^{d_1 \times d_2}, ..., W_L \in \mathbb{R}^{d_{L-1} \times d_L}$, denotes the weights across L hidden layers, respectively. [4] Mathematically, training (optimizing) a neural net can be described as follows. Let $\boldsymbol{W}^*$ denote the optimal weights for the following ERM over a set of samples $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} L_{\boldsymbol{S}}(\boldsymbol{W}) = \arg\min_{\boldsymbol{W}} \frac{1}{n} \sum_{i=1}^{n} l(\mathcal{N}_{\boldsymbol{W}}(x_i), y_i) \tag{8}$$

for some loss function $l(\cdot, \cdot)$. For simplicity, we assume $d = n$ and the feature set $X \in \mathbb{R}^{n \times d}$ of original training data is invertible. With a similar reasoning, we can instead feed $\boldsymbol{I}_d$, the $d \times d$ identity matrix, to train an L-layer fully-connected neural net to obtain the weights $\boldsymbol{W_I} = (W_1, W_2, ..., W_L)$. Consider a new neural net, where the weight of the first layer is changed to $\boldsymbol{W}_X = (X^{-1}W_1, W_2, ..., W_L)$. Note that the loss of $\mathcal{N}_{\boldsymbol{W_I}}$ w.r.t. $\boldsymbol{I}_d$ is equal to that of $\mathcal{N}_{\boldsymbol{W}_X}$ w.r.t. X in (8). Also, it is clear that the training to obtain $\boldsymbol{W_I}$ does not require any knowledge of X, so perfect secrecy is preserved.

Though there is a bijective relationship between the $\boldsymbol{W}_I$ and $\boldsymbol{W}_X$, we have to remark that in deep learning with non-convex activation functions, the number of local minima increases exponentially as the number of neurons and dimensionality increases [AHW+96]. This brings significant efficiency concerns that whether after transformation, training through classic optimization tools such as (Stochastic) Gradient Descent ((S)GD) can find good models.

3 Random Transformation and Information Theoretical Privacy

In Section 2, we focused on the first type of transformation, which is data-dependent but with perfect secrecy. In the following, we will focus on the second type, where T is randomly selected from some predefined function set C_T. Such a transformation is more smooth and will preserve certain dependence between input data and transformed output. This will allow us to handle a much wider class of learning problems with generalized privacy notions.

We first define three privacy interpretations in Section 3.1, give an overview of our strategy for PAC inference resistance in Section 3.2, and present our protocol in Section 3.3.

3.1 Information Theoretical Privacy

Additional Information: A primary strategy in both cryptography and information theoretical security is to measure how much *additional* information (knowledge) is provided by the output of a mechanism. It is expected that no matter what prior knowledge is assumed for the adversary, after observing outputs of

[4] For the case where each layer is defined as $\sigma(x \cdot W + b)$ with an additional addition term b, one may replace $\hat{x} = (x, 1)$ in (7), which does not lose generality.

a privacy-preserving mechanism, the difference between the prior and the posterior belief w.r.t. the input should be limited. The first formal definition, perfect secrecy, was proposed by Shannon in [Sha49], where a mechanism $\mathcal{M}$ is perfectly secure if for arbitrary output o, the likelihood $\mathcal{L}(o|I) = \Pr(\mathcal{M}(I) = o)$ should be identical for any possible input I. In the previous section, we provided a concrete example to achieve perfect secrecy: a protocol always outputs a constant identity matrix independent of inputs. Perfect secrecy is ideal but costly (or limited): information is not propagated under a perfectly secure mechanism. One natural generalization is to quantify such additional information with well-defined metrics. Inspired by the ideas of K-anonymity [Swe02], we introduce the following notion.

Ambiguity Set Towards more interesting tradeoffs, one strategy is to preserve identical likelihoods within a subset of the input domain rather than globally. We first formalize the definition of *ambiguity set*.

Definition 2 (Ambiguity Set). *For a mechanism $\mathcal{M}$, we call S an ambiguity set of $\mathcal{M}$ if for any elements $X_1, X_2 \in S$, the distribution of outputs $\mathcal{M}(X_1)$ and $\mathcal{M}(X_2)$ are the same.*

Another interpretation of the above definition is through a view of indistinguishability.

Definition 3 (Indistinguishability). *For a mechanism $\mathcal{M}$, with respect to any possible output o, two inputs X_1 and X_2 are indistinguishable if the likelihood* $\Pr(\mathcal{M}(X_1) = o) = \Pr(\mathcal{M}(X_2) = o)$. [5]

Thus, any two elements within an ambiguity set are indistinguishable with respect to any outputs. Said another way, elements within an ambiguity set produce outputs in the same distribution [6]. To provide an example, we set the transformation T to be linear on an input matrix X as $T(X) = XA$ for some matrix A.

Lemma 3. *If A is a zero-mean Gaussian matrix, of which each entry is i.i.d. generated by a zero-mean Gaussian distribution, then such a random linear transformation provides an ambiguity set w.r.t. X which includes all matrices X' such that $X' = X \cdot U$ for some unitary matrix.*

Proof. To avoid a tedious discussion on the support domain of the distribution of XA, we provide a proof as follows. Since each column of A is independent, we consider the first column of A such that $XA(:, 1) = o$ for some o. Then, we have an identity that $XU \cdot U^{-1}A(:, 1) = o$ for some unitary matrix U. On the other hand, the distribution density of a zero-mean Gaussian vector $A(:, 1)$ is proportional to $exp(-\sum_{i=1}^{d} A(i, 1)^2) = exp(-\|A(:, 1)\|^2)$. It is noted that U^{-1} is still unitary, where $\|U^{-1}A(:, 1)\| = \|A(:, 1)\|$ is preserved. Thus, for $X' = XU$, the distribution of XA and XA' is identical.

In our context, if $X \in \mathbb{R}^{n \times d}$, where each row represents a sample, under the above transformation, one cannot distinguish two sets of n samples whose respective pairwise distances are the same. In Appendix B, we give another example of how a linear transformation can generate an ambiguity set as the column space of its input matrix.

[5] Here, we abuse a bit the notion $\Pr(\cdot)$, which denotes the probability density if the distribution is continuous.

[6] One important security issue is the composition of information leakage from multiple observations of a mechanism on one private dataset. Ambiguity set is resistant to composition leakage: the ambiguity set is maintained regardless of multiple arbitrary calls to $\mathcal{M}$ on one input dataset X.

Distribution Divergence (Differential Privacy) Another direction is to consider the difference between the output distributions generated by two inputs. A representative approach is Differential Privacy (DP). At a high level, we call an algorithm differentially private if from its output, it is hard to distinguish the participation of an individual sample. In the following, we generalize the idea of (ϵ, δ)-DP to quantify such indistinguishability as follows:

Definition 4 ($(\epsilon, \delta, d, \tau)$-Privacy). *A mechanism $\mathcal{M}$ is $(\epsilon, \delta, d, \tau)$ private if for any two inputs X_1 and X_2 such that under metric d, $d(X_1, X_2) \leq \tau$, and for any possible output set S,*

$$\Pr(\mathcal{M}(X_1) \in S) \leq e^{\epsilon} \Pr(\mathcal{M}(X_2) \in S) + \delta.$$

Such a definition measures the difference between the likelihoods of two close enough X_1 and X_2 ($d(X_1, X_2) \leq \tau$) to produce an output o in the worst case. This can be interpreted as a tradeoff between type I and II error in a view of hypothesis testing [DRS19]. If we select d to be the Hamming distance and $\tau = 1$, then it becomes (ϵ, δ)-DP.

Definition 5 ((ϵ, δ)-DP [DMNS06]). *A mechanism $\mathcal{M}$ is (ϵ, δ) differentially private if for any two adjacent datasets X_1 and X_2, which differ only in one sample, and any possible output set S,*

$$\Pr(\mathcal{M}(X_1) \in S) \leq e^{\epsilon} \Pr(\mathcal{M}(X_2) \in S) + \delta.$$

Many variants can be easily derived with other selections of the metrics to measure the difference between the distributions of $\mathcal{M}(X_1)$ and $\mathcal{M}(X_2)$, for example, KL [WLF16] or α-Renyi [Mir17] divergence.

Data Recovery: As a short summary, a mechanism with ambiguity set or DP guarantees indicates that there is no additional information provided by the outputs w.r.t. the inputs within one ambiguity set, or the additional information is limited and quantified by the parameters (ϵ, δ) for inputs close enough to each other (sensitivity restriction). Though rigorous metrics are provided, the above two privacy notions still fail to answer one of the most intuitive questions about privacy leakage: how much information an adversary can recover w.r.t. the private data given an observation. Therefore, we propose another line to define privacy by the performance of adversary approximation of the private input. To answer how much information is leaked, we typically need more specific assumptions on the prior knowledge, which is also necessary in this context. One can imagine an extreme case where an adversary already has the full knowledge of the true input and thus a perfect recovery can always be produced regardless of any mechanism or observations.

Probably Approximately Correct (PAC) Approximation We mimic PAC learning theory to set the privacy metric in a form that given observations, can an adversary approximate, with error smaller than ϵ, the true input with confidence at least $(1 - \delta)$? More formally,

Definition 6 (Resistance to (ϵ, δ)-PAC Approximation). *We call a mechanism $\mathcal{M}$ resistant to (ϵ, δ)-PAC approximation for private input data X in a distribution P, if given the output $\mathcal{M}(X)$, there does not exist an (possibly randomized) estimator $g(\cdot)$ based on $\mathcal{M}(X)$ such that*

$$\Pr_{X \sim P}(\|g(\mathcal{M}(X)) - X\| < \epsilon) \geq 1 - \delta. \tag{9}$$

In the above definition, the data distribution P captures the prior knowledge from the adversary's view. Also, it is notable that in Definition 6, which we abbreviate PAC security, no restriction is put on the estimator g. In some sense, PAC security is a more comprehensive characterization of data security/privacy compared

to merely additional information measurement. That is, an adversarial recovery is determined both by the additional information provided by ciphertext and the adversary's prior knowledge on plaintext. With the above preparation, we now proceed to describe how to construct an information theoretical security amplification via both data mixing and transformation.

3.2 Overview of Strategy

To begin, we formalize the knowledge of an adversary w.r.t. the training dataset. For a training dataset $S = \{s_1, s_2, ..., s_n\}$ of size n, we assume that an adversary has complete knowledge of k samples $\{s_1, s_2, ..., s_k\}$ within S, termed the public part of S, and denoted $S_A = \{s_{[1:k]}\}$. The remaining $(n-k)$ samples $\{s_{[n-k+1:n]}\}$, where the adversary may have some prior but not full knowledge, are termed the private part of S, and denoted $S_P = \{s_{[k+1:n]}\}$.

As in the information flow shown in Fig. 1, privacy leakage is due to the exposure of the transformed data $T(S) = \{T(s_1), T(s_2), ..., T(s_n)\}$[7], because the subsequent training procedure is merely postprocessing. We will model the adversary inference to recover the private $S_P = \{s_{[k+1:n]}\}$ given the observation set $T(S) = \{T(s_1), T(s_2), ..., T(s_n)\}$ and the public part $S_A = \{s_{[1:k]}\}$.

After quantifying the power of the adversary and user by the number of samples that each has access to, we now turn to characterize the selection of C_T in two ways. On the utility side, it is hoped that the generalization error of transformed data described below is not compromised significantly:

$$\min_{f \in C_f} \mathbb{E}_{\boldsymbol{s} \sim P^n, T \sim C_T} l[f(T(\boldsymbol{s}))], \tag{10}$$

where we use P^n to denote the joint distribution of n i.i.d. samples and C_f as the candidate model set to be optimized over. On the other hand, C_T is set to maximize the adversary inference error:

$$\max_{C_T} \Pr_{T \sim C_T} \big(\|Adv\big(T(s)\big) - s\| > \epsilon, s \in S_P \mid S_A, T(S) \big), \tag{11}$$

where Adv denotes an arbitrary inference/learning algorithm that an adversary can apply to invert the transformed private part S_P. Note that the distribution of elements in S_A may not necessarily equal the global sample distribution P. For example, the adversary may only know a specific class of samples used in a classification task.

As both the user and the adversary will address their own learning task, before proceeding, let us first provide a rough estimation of their respective expected learning performances. Described in a Probably Approximately Correct (PAC) model, we assume that the true data generation model of S is from a function (*concept*) class C_o, i.e., the sample $s = (x, y)$ is generated in a form $y = h(x)$ for some $h \in C_o$. For simplicity, we temporarily assume both C_o and C_T are finite function sets. In a PAC setup, given samples, the goal of a learner is to determine the true function/concept h from the candidate set C_o, where the performance metric is defined as the error rate that one outputs a wrong function $h' \neq h$. Here, we denote the Vapnik-Chervonenkis (VC) dimension [BEHW89] of C_o as $VC(C_o)$. Intuitively, $VC(C_o)$ captures the hardness of the learning task. After transformation, the function class now becomes the Cartesian product of C_T and C_o, i.e., $C_T \times C_o$. It is not hard to prove that the VC dimension of the product of two function classes is upper bounded by the product of the VC dimension of each. Here, we consider the worst case and approximate the VC dimension of $C_T \times C_0$ by $VC(C_T) \cdot VC(C_0)$. Since the user holds n samples, we know the error rate is $\Theta(\frac{VC(C_T)VC(C_0)}{n})$ [AD96].

[7] In the following, without loss of generality, we assume the transformation may be applied on both feature domain $\mathcal{X}$ and the label domain $\mathcal{Y}$, and denote the transformed sample as $T(s)$.

With a similar reasoning, adversary inference to approximate the inverse of T can also be described by a PAC model. In this learning task, take the feature as $T(s)$ and label as s. If C_T is formed by invertible functions, the data generation is in a form $s = T^{-1}(T(s))$. Clearly, the corresponding concept set $C_{T^{-1}}$ of the inverse transform T^{-1} is bijective to C_T, where determining the true inverse of T is equivalent to determining T itself. Thus, even with the restriction that the adversary only knows S_A and $T(S_A) = \{T(s_1), T(s_2), ..., T(s_k)\}$ instead of $T(S)$, the error rate is $\Theta(\frac{VC(C_T)}{k})$. Naturally, the relationship between utility and privacy appears as a tradeoff in terms of $VC(C_T)$: a larger $VC(C_T)$ implies a potentially better data obfuscation but also a greater utility loss. However, the advantage rate, i.e., how much more the adversary has to pay compared to the cost of a user, is merely linear in the ratio n/k, and is far from a trapdoor. *Thus far, we have only resorted to additional information loss to preserve privacy.* To amplify the advantage and force an *unfair game* between adversary and user, we need to further explore underlying uncertainty in data.

Subsampling and Entropy of Private Data: First, a natural resource is permutation. A particular sequence of different samples is in general not needed in the learning procedure. Shuffled transformed data has great potential to incur an exponential computational complexity for an adversary to approximate the correct correspondence between $s_{[1:k]}$ and $T(s_{[1:k]})$ before any meaningful inference. But permutation is not sufficient to show information theoretical hardness.

With a closer look at the earlier analysis, note that we restricted the transformation on the training dataset such that each sample in either S_A or S_P is separately transformed. Though the fundamental structure of our framework is *multiple encryptions with the same key*, we can force adversary inference *without full knowledge of plaintext* and *learning with errors*. However, the trick here is we do not plan to introduce real noise but utilize the *entropy of private data* S_P. [8]

To utilize the entropy of private data as an obfuscation, a straightforward approach is to mix public and private original data before the uniform transformation T. Indeed, data augmentation provides various candidates for a mix strategy. Roughly speaking, data augmentation is a large class of methods to modify or reconstruct training data, which may improve the generalization and robustness especially in deep learning. Most data augmentation approaches are heuristic but widely applied in practical training.[9] In the following, we take *mixup* [ZCDLP18], which has received considerable attention recently, as a concrete example.

Mixup can be simply described by two steps. Provided n samples $S = \{s_1, s_2, ..., s_n\}$, where $s_i = (x_i, y_i)$, *mixup* first randomly subsamples, say $s_{i_1}, s_{i_2} \in S$, and a random weight λ within $(0, 1)$. Then, a new virtual sample $\bar{s} = (\bar{x}, \bar{y})$ is defined by the natural weighted average of s_{i_1}, s_{i_2} with λ:

$$\bar{x} = \lambda x_{i_1} + (1-\lambda)x_{i_2}, \qquad \bar{y} = \lambda y_{i_1} + (1-\lambda)y_{i_2}. \tag{12}$$

Imagine the case that we apply *mixup* on S_A and S_P such that s_{i_1} and s_{i_2} are randomly selected from S_A and S_P, respectively. Hence, even with the prior knowledge on S_A, the adversary now loses the full knowledge of sample generation: she now only has partial information of the preimage of transformed data. We formalize adversary inference under the *mixup* setup in Algorithm 2, termed *Learning with Incomplete Labels*.

To quantify the information theoretical hardness of adversary inference, we set out to lower bound the sample complexity that one needs to conduct any efficient inference. Assume that an adversary has access to the oracle for m queries and Adv to be an algorithm (can be randomized) which will output some approximation

[8] The computational hardness of, for example, Ring Learning with Errors must count on its finite algebra structure. If the oracle is over the real domain, many statistical or optimization approaches can break it easily.

[9] These heuristic data reconstructions are indeed appealing to private learning, where randomization introduced can amplify privacy but without utility compromise or even strengthening the performance.

Algorithm 2 Learning with Incomplete Labels by Adversary

1: Adversary queries for a new sample.
2: Oracle randomly selects $s_a \in S_a$ and $s_p \in S_p$ and applies a *mixup* protocol $\mathcal{MIX}$ on (s_a, s_p) and returns $\tilde{s} = (s_a, T\big(\mathcal{MIX}(s_a, s_p)\big)$.
3: Adversary approximates the inverse of T where we quantify the security level as

$$\Pr(\|Adv(T(\hat{s})) - \hat{s}\| < \epsilon) < \delta$$

for sample $\hat{s} \sim P$.

with required performance dependent on the m observations, denoted by $\tilde{s}^m = \big(s_a^m, T\big(\mathcal{MIX}(s_a, s_p)\big)^m\big)$.[10] Let us consider the mutual information between T and Adv [GH01]:

$$I(T; Adv) \leq I(T; \tilde{s}^m) = mI(T; \tilde{s}). \tag{13}$$

In the following, we use $I(a; b)$ to denote the mutual information between two variables a and b and $H(a)$ to denote the entropy where $I(a, b) = H(a) - H(a|b)$. Here, the first inequality in (13) is because Adv is a function of observations $\tilde{s}^m$ and the second equality is because we assume the mixed sample is generated independently. This renders a lower bound on the sample complexity, namely,

$$m \geq \frac{I(T; Adv)}{I(T; \tilde{s})}. \tag{14}$$

(14) coincides with our intuition: the numerator represents how much information is needed for required approximation error with satisfied confidence, while the denominator represents how much information is provided per mixed sample on average. Detailing $I(T; \tilde{s})$, we have

$$\begin{aligned} I(T; \tilde{s}) = I\big(T; s_a, T\big(\mathcal{MIX}(s_a, s_p)\big)\big) &= I\big(T; s_a\big) + I\big(T; T\big(\mathcal{MIX}(s_a, s_p)\big)|s_a\big) \\ &= I\big(T; s_a\big) + H(T|s_a) - H\big(T|s_a, T\big(\mathcal{MIX}(s_a, s_p)\big)\big). \end{aligned} \tag{15}$$

As a comparison, let us consider the case without data mix and the transformation T separately applied to each sample. It is noted that the sample bound (14) on m still holds, whereas the observation $\tilde{s}$ becomes simpler, namely, $(s_a, T(s_a))$. In such a case, the denominator $I(T; \tilde{s})$ is

$$\begin{aligned} I(T; \tilde{s}) = I(T; s_a, T(s_a)) &= I(T; s_a) + I(T; T(s_a)|s_a) \\ &= I(T; s_a) + H(T|s_a) - H(T|s_a, T(s_a)). \end{aligned} \tag{16}$$

Comparing (16) with (15), since mixed data is more fuzzy, provided the transformed data, the differing term, the conditional entropy $H\big(T|s_a, T\big(\mathcal{MIX}(s_a, s_p)\big)\big)$ should be larger than $H\big(T|T(s_a), s_a\big)$, which implies better obfuscation and a larger sample complexity will be required under data mix.

We have to stress that the above analysis is based on the view of an adversary, where the entropy of private data is determined by her prior knowledge. When the private sample s_p and the weight λ both become constant, there will be no difference between (16) and (15). Though entropy is a relative notion, in practice, especially in computer vision, the great diversity of large datasets implies a huge amount of entropy under reasonable assumptions. More formal analysis will be provided in Section 4.3 along with concrete lower bounds on the sample complexity.

[10] These observations include the correspondence between the public sample and the resultant mixed sample, which is not exposed to the adversary in the Dauntless framework, but could conceivably be inferred through prior knowledge.

3.3 Algorithm Description

We now flesh out the main protocol as Algorithm 3.

Algorithm 3 Main Protocol in Dauntless Framework: Data Mix and Random Transformation

Input: Training Dataset of n samples $S = s_{[1:n]} = \big((x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\big)$ with feature $x_i \in \mathbb{R}^{1\times d}$ and label $y_i \in [1 : c] = \{1, 2, ..., c\}$ for c classes. $S = S_A \cup S_P$.

Phase 1 - Random Label Encoding

1: Randomly generate $\bar{Y} \in \mathbb{R}^{c\times c'}$, for some $c' \geq c$, such that the c rows of $\bar{Y}$ are (almost) orthogonal to each other.
2: For each sample $s_i = (x_i, y_i)$, replace the label y_i by the y_i-th row of $\bar{Y}(y_i, :)$.

Phase 2 - Data Mix

1: **for** $j = 1 : m$ **do**
2: Independently and uniformly sample $s_a \in S_A$ and $s_p \in S_P$, respectively.
3: Randomly generate $\lambda \in (0, 1)$.
4: Reconstruct mixed sample $\bar{s}_j = \lambda s_a + (1 - \lambda)s_p$, with the weighted average on both feature and label.
5: **end for**

Phase 3 - Transformation

1: Randomly generate L weight matrices $W_1 \in \mathbb{R}^{d\times d_1}, W_2 \in \mathbb{R}^{d_1\times d_2}, ..., W_L \in \mathbb{R}^{d_{L-1}\times d_L}$, in which each entry is independently generated from a distribution Q.
2: Construct a neural network structure of L hidden layers with an activation function σ and weights $W_{[1:L]}$: $\mathcal{N}_{\boldsymbol{W}}(x) = \sigma(..., \sigma(\sigma(x \cdot W_1) \cdot W_2), ..., W_L)$.
3: For each mixed sample $\bar{s}_j = (\bar{x}_j, \bar{y}_j)$, compute the transformation $T(\bar{s}_j)$ as $\big(\mathcal{N}_W(\bar{x}_j), \bar{y}_j\big)$.

Output: m transformed samples $\big(\mathcal{N}_W(\bar{x}_j), \bar{y}_j\big), j = 1, 2, ..., m$.

The data processing on the user side can be summarized in three steps, which is formally presented in Algorithm 3. The first step is to encode the label. Please note in *mixup* [ZCDLP18], the labels are encoded as one-hot vectors.[11] For example, in a binary classification, we have a picture x_1 of a cat and a picture x_2 of a dog, where $(1, 0)$ denotes the cat label and $(0, 1)$ the dog. When one mixes the two pictures with weights 0.3 and 0.7, respectively, the feature of the mixed sample becomes $0.3x_1 + 0.7x_2$ with the corresponding label as $(0.3, 0.7)$. From a privacy perspective, one-hot vector encoding exposes the random weight used in the data mix. To avoid this, in Phase 1, we randomly generate a set of (almost) orthogonal vectors to alternatively encode the label, which can be viewed as a random transformation on the label domain.

The second step is a straightforward data mix, which has been formally discussed before. Public data known by the adversary and private data are mixed into m new samples. Finally, the last step is applying a random transformation over the mixed feature domain. Here, we select the function in a form of an L-layer neural network with random weights.

With the sample transformation procedure described in Algorithm 3, the transformed samples will then be sent to the server, who is expected to train a model f_T over $\big(\mathcal{N}_W(\bar{x}_j), \bar{y}_j\big), j = 1, 2, ..., m$. With the knowledge of transformation T and label encoding bases $\bar{Y}$, the user can then conduct (private) inference on new samples via f_T, as described in Algorithm 4 in Section 6.

[11] Generally speaking, in *mixup* we cannot take the label as a real number unless the learning task can be addressed by a regression.

4 Privacy Analysis

Given the three privacy notions defined in Section 3.1, we will study the privacy guarantees provided by Algorithm 3 from the three perspectives.

To give a high-level picture, consider a random transformation $T(X, \theta)$ on dataset X, where $T(\cdot, \theta) : \mathbb{R}^{n\times d} \to \mathbb{R}^n$ with randomness seeds θ. Given X, if T is bijective w.r.t. θ, then for output $z = (z_1, z_2, ..., z_n)^T = T(X, \theta)$, we have

$$\mathcal{P}_z(z) = \mathcal{P}_\theta(\theta = T^{-1}(X, z))|det(\partial_z T^{-1}(X, z))|, \tag{17}$$

where the Jacobian $\partial_z T^{-1}(x, z) \equiv \partial T_i^{-1}(x, z)/\partial z_j$. Here, $T^{-1}(X, z)$ denotes the unique inverse of the random seeds θ such that $T(X, T^{-1}(X, z)) = z$. Recall the definitions in Section 3.1, if two data matrices X_1 and X_2 are within an ambiguity set, then the outputs $T(X_1, \theta)$ and $T(X_2, \theta)$ are distributed identically. Similarly, substituting the density function (17) into Definition 4, one can derive corresponding DP parameters. We flesh out the privacy analysis throughout this section.

4.1 Ambiguity

The ambiguity set of the sample transformation described in Algorithm 3 should be characterized by the resultant ambiguity from both the feature and label transformation. On the feature side, two operations, data mixing (Phase 2 in Algorithm 3) and uniform transformation (Phase 3 in Algorithm 3) are applied on the feature set X. The resultant ambiguity should be at least that which is produced by one operation on them. Therefore, according to Lemma 3, if the weights of the uniform transformation T are i.i.d. generated by a zero-mean Gaussian, then Algorithm 3 produces ambiguity w.r.t. X that one cannot distinguish two feature sets which have the same respective pairwise distance.

For the ambiguity w.r.t. the labels, we shift our focus to the private weights applied during mixing. We first describe the label encoding (Phase 1 in Algorithm 3) and the label mixing (Phase 2 in Algorithm 3) in a matrix form. There are many ways to generate (almost) orthogonal vectors. Here, we consider $\bar{Y}$ to be a randomly generated zero-mean Gaussian matrix in a form $Y = [v_1^T, v_2^T, ..., v_c^T]^T$ as the label bases. Each v_i represents a label for a class. Thus, equivalently we can represent the original training dataset $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ by $\{(x_1, v_{y_1}), (x_2, v_{y_2}), ..., (x_n, v_{y_n})\}$. Note that, during the data mixing (Phase 2), every mixed sample is a linear combination of the two original samples selected. Therefore, we can use a matrix M to describe the mixing procedure. Let (j_1, j_2) be the indices and $(\lambda_j, 1-\lambda_j)$ be the corresponding weights used of original samples to be mixed for the j-th mixing with $j = 1, 2, ..., m$, respectively. Then, the matrix M is in a form where the j-th row of M, i.e., $M(j, :)$, is all zeroes except for the y_{j_1}-th and y_{j_2}-th entries, the two classes of labels to be mixed, which are λ_j and $(1 - \lambda_j)$, respectively. Therefore, the encoded label $\tilde{Y}$ of mixed data can be written as,

$$\tilde{Y} = M \cdot Y. \tag{18}$$

Clearly, the mixing matrix $M \in \mathbb{R}^{m\times c}$ is stochastic, where the sum of each row equals 1. The following theorem presents an ambiguity set w.r.t. M.

Theorem 1. *Algorithm 3 renders an ambiguity set w.r.t. the private mixing matrix M where two mixing matrices M and M' belong to one ambiguity set if $M = UM'$ for some unitary matrix U.*

Proof. We first rewrite (18) as the following identity,

$$\tilde{Y} = M \cdot V \cdot V^{-1} \cdot Y. \tag{19}$$

Here, V is an arbitrary invertible matrix. It is noted that the label bases Y is randomly generated by i.i.d. normal distribution, of which the density function should be proportional to $exp(\sum_{i=1}^{c}\sum_{j=1}^{c} Y(i,j)^2)$, where $Y(i,j)$ denotes the entry of Y at the crossing of the i-th row and j-th column. Since a unitary matrix will preserve the l_2-norm of the vector mapped, UY and Y enjoy the same probability density. On the other hand, if two mixing matrices M and M' have the relationship as $M' = M \cdot V$ for some unitary matrix V, then substitute $M' = M \cdot V$ and $Y' = V^{-1} \cdot Y$ into (19), and the two pairs (M, Y) and (M', Y') produce the same $\tilde{Y}$ with the same likelihood.

As a corollary, note that in Algorithm 3, we only mix two samples each time. If we do not restrict the mixing matrix M but randomly select M to be a stochastic matrix, then the ambiguity set can be further described as follows:

Corollary 1. *If the mixing matrix M is randomly selected to be a stochastic matrix, then in the above label encoding and mixing procedure, every M enjoys an ambiguity set which includes all elements in a form $M \cdot U$, where U is some stochastic unitary matrix.*

4.2 Differential Privacy

In this section, we turn to analyze the obfuscation from a statistical divergence or DP perspective. In such a view, an algorithm is more private if the output of an algorithm is less sensitive to the input, which implies more *information loss* during the transform. Recall in Section 3.2, we used the VC dimension to interpret the tradeoff between privacy and utility of a transformation. The results presented in this subsection can supplement that analysis by quantifying the dependence between input and output.

DP mechanisms have been widely studied wherein the most common tool is additive noise such as Gaussian or Laplace Mechanism [BST14]. However, in our context, both the uniform transformation T and data mixing can be viewed as an *multiplicative* operator on the data. We take the feature dataset X as an example. In Algorithm 3, recall the analysis shown in Section 4.1, the mixing procedure can be described by a left-multiplication matrix M as $M \cdot X$, where M represents the mixing weight. On the other hand, the uniform transformation T can be captured by a (generalized) right-side linear transformation. For each layer, the neural network is in a form $\sigma(X \cdot W)$, where W and σ represent the weight and activation function, respectively. This raises an interesting question: what kind of DP guarantee can be provided by a multiplicative mechanism? Of independent interest, [BBDS12], [KKMM13] study a related problem that through random projection, how the Johnson–Lindenstrauss (JL) Lemma preserves DP. [12] In this section, we set out to provide a more systematic study on this topic. Without loss of generality, in the following, we restrict our focus to what DP guarantees are achievable by a linear transformation $X \cdot A$ w.r.t. the private input matrix $X \in \mathbb{R}^{n \times d}$.

First, to avoid some tedious discussion, we assume some normalization on input X. It is noted that after any linear transformation, XA is always within a linear space spanned by the columns of X (column space). Therefore, for non-trivial DP guarantees, we have to restrict the private inputs X to share a same column space. To overcome the restriction, we assume a normalization processing step on X which expands X to be an invertible square matrix $\mathbb{R}^{n \times n}$ if $n > d$. A simple expansion is that we duplicate X in a concatenated form $[X|X|...|X]$ until it becomes a square matrix. If n cannot be divided by d, we randomly select $(n \mod d)$ columns from X to be the last copy. In Section 2, we mentioned two approaches to make a matrix invertible, such as diagonal perturbation and SVI. Thus, to ease presentation, in the following, we just assume X to be a $n \times n$ invertible matrix.

[12] JL Lemma states that there exists a dimensional reduction mapping which preserves pairwise distance of elements within a set with high probability. JL Lemma has a close relationship to compressive sensing and locality-sensitive hashing. In our context, the uniform transformation T is not necessarily designed to preserve the pairwise distance.

In additive DP mechanisms, it is well known that (Laplace/Gaussian) noise of greater variance produces stronger DP guarantees (smaller ϵ, δ under the same setup). However, multiplicative DP does not match this intuition, i.e., a linear operator A generated by a more heavy-tailed distribution does not simply strengthen DP in general. One of the main reasons is that the *multiplicative sensitivity* depends not only on the input difference but also on the singularity of X. (We include an interesting result for the special case if X is diagonal in Appendix C.) In general, for two invertible matrices $X_1, X_2 \in \mathbb{R}^{n\times n}$ where $E = X_2 - X_1$ denotes their difference, the transformed distribution of $X_1\boldsymbol{\alpha}$ and $X_2\boldsymbol{\alpha}$ for some random vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ in distribution $\mathcal{Q}$, is equivalent to studying

$$\frac{\mathcal{P}(X_1\boldsymbol{\alpha} = \boldsymbol{z})}{\mathcal{P}(X_2\boldsymbol{\alpha} = \boldsymbol{z})} = \frac{det(X_2)}{det(X_1)} \cdot \frac{\mathcal{P}(X_1^{-1}\boldsymbol{z})}{\mathcal{P}(X_2^{-1}\boldsymbol{z})} = \frac{det(X_1 + E)}{det(X_1)} \cdot \frac{\mathcal{P}(X_1^{-1}\boldsymbol{z})}{\mathcal{P}((X_1 + E)^{-1}\boldsymbol{z})}. \tag{20}$$

Replacing $\boldsymbol{z}$ by $(X_1 + E)\boldsymbol{\alpha}$ in (20), it is equivalent to considering $\frac{det(X_1+E)}{det(X_1)} \cdot \frac{\mathcal{P}(\boldsymbol{\alpha}+X_1^{-1}E\boldsymbol{\alpha})}{\mathcal{P}(\boldsymbol{\alpha})}$. Thus, a natural question raised is how to select the distribution of $\boldsymbol{\alpha}$ to minimize the ratio.

We provide a framework to study this problem. Without loss of generality, we assume $\mathcal{P}(\boldsymbol{\alpha}) \propto e^{\psi(\|\boldsymbol{\alpha}\|^2)}$ for some function $\psi(\cdot) : \mathbb{R} \to \mathbb{R}$ such that $\int_{\mathbb{R}^n} e^{\psi(\|\boldsymbol{\alpha}\|^2)} d\boldsymbol{\alpha} < +\infty$. In addition, we assume the Lipschitz constant of $\psi(\cdot)$ is L in the following, i.e., $|\psi(a) - \psi(b)| \leq L|b - a|$. Clearly, to control the difference quantity $\|X^{-1}E\boldsymbol{\alpha}\|$, we need proper assumptions on the norms of E and the tail bound of $\|\boldsymbol{\alpha}\|$. Following the idea in [BBDS12], to avoid any further restrictions on the data matrix X, when X is singular or close to being singular, we will perform the following modification before transformation. Similar to SVI, assume that the SVD of X is $X = U\Sigma V$. We modify X to $\hat{X} = U(\sigma + \tau \cdot \boldsymbol{I})V$, thereby the least singular value of $\hat{X}$ is at least τ [13].

Theorem 2. *Let $P(\boldsymbol{\alpha}) \propto e^{\psi(\|\boldsymbol{\alpha}\|^2)}$, where $\Pr(\|\boldsymbol{\alpha}\| > r) < \delta_r$. For a random matrix $W \in \mathbb{R}^{n\times d'}$ such that each row of W i.i.d. follows the distribution of $\boldsymbol{\alpha}$, the following transformation on private data matrix, $X \in \mathbb{R}^{n\times n}$, namely, $X \to \hat{X} = U(\sigma + \tau \cdot \boldsymbol{I})V \to \hat{X}W$, satisfies (ϵ, δ)-DP, where*

$$\epsilon = d'\epsilon_0(e^{\epsilon_0} - 1) + \epsilon_0\sqrt{2d' \log(1/\tilde{\delta})},$$

and

$$\delta = d'(\delta_0 + \delta_r) + \tilde{\delta}.$$

Here,

$$\epsilon_0 = \frac{\mathcal{B}}{\tau} + 4L\frac{\mathcal{B}}{\tau}(1 + \frac{\mathcal{B}}{\tau})\frac{-r^2\log(\delta_0/32)}{n + 2\sqrt{n\log(\delta_0/16)}},$$

and δ_0 and $\tilde{\delta}$ are free parameters to be selected. The sensitivity parameter $\mathcal{B}$ is defined as the upper bound of the l_2 norm of the difference between any two samples.

The proof of Theorem 2 can be found in Appendix D. Taking Theorem 2 as the building block, one can further characterize the DP guarantees produced by a multi-layer neural network with random weights. Equipped with non-linear activation functions, for example, the sigmoid function, the sensitivity may contract when data passes through more layers. From Theorem 2, ϵ is proportional to the ratio $\mathcal{B}/\tau$, while, in our case, a larger τ implies a heavier regularization on the data and may compromise the performance. From our experiments on MNIST and CIFAR-10, we find that if we set $d' = d$ and the sensitivity $\mathcal{B}$ in a scale of $1/d$, with a Gaussian matrix and proper selection of τ to produce $\epsilon = 1$, there is no obvious performance drop.

[13] SVI is not the only way to improve the sensitivity. Ideally, in the learning after transformation context, it is expected that we can apply some uniform transformation on the data matrix to increase the small singular values.

4.3 PAC Security

In Section 3.1, we formally introduced the PAC security definition and in Section 3.2, we provided a generic framework to lower bound the sample complexity for required estimation performance. In this section, we follow the strategy to concretely connect the PAC security budget, i.e., the (ϵ, δ) inference restriction given in (9), with the sample complexity in two important scenarios: First, towards Corollary 2, we continue the focus on the PAC security of data, which captures how much information can be recovered w.r.t. private samples; Second, we change the security objective to be the transformation T itself and study how (statistically) closely an adversary can approximate T in Theorem 5. PAC security studied in the above two cases will be the building blocks to characterize the privacy in two dimensions: the data privacy loss during model training and later model applications, and defending against model extraction (stealing) through multiple accesses [CJM20], which we will explore in Section 6.

PAC Security of Samples We flesh out the analysis of adversarial inference with specification on the data and transformation. To ease the analysis, we assume the private mixing weights are known to the adversary, as clarifying below, and we instead focus on the PAC security of the feature: For the data, we assume every mixed sample is formed by a private sample and a public sample, which are random Gaussian vectors: $s_a, s_p \in \mathbb{R}^{1\times d}$ and every entry of s_a and s_p independently follows a Gaussian distribution $N(0, \tau_a)$ and $N(0, \tau_p)$, respectively. [14] We assume the mix weight to be fixed as $(1, 1)$ (without normalization). As shown in (14) and (15), compared to fixed weights, random weights will always lead to a smaller $I(T; \hat{s})$, the average information provided by a single sample, which implies better privacy. Following the definition of PAC approximation in Section 3.1, for a mixed sample $\bar{s}$ generated by $s_a + s_p$ with the transformation T, a mechanism $\mathcal{M}_{(s_a,\bar{s})^m}$, based on m observations denoted by $(s_a, T(\bar{s}))^m$, can return an *ϵ-approximation to the inverse of* T^{-1} (if it exists) to recover the input with confidence $(1-\delta)$, if

$$\Pr_{\bar{s}\sim s_a+s_p, T}(\|\mathcal{M}_{(s_a,T(\bar{s}))^m}(T(\bar{s})) - \bar{s}\| < \epsilon) \geq 1-\delta. \tag{21}$$

Note that in the observations, we are conservatively assuming that the adversary knows the correspondence between each mixed sample and the public sample that was used to generate it. Further, in this subsection, we assume T is a one layer neural network with some activation function σ, i.e., $T(\bar{s}) = \sigma(\bar{s}W)$. We will assume the weight matrix W to be an invertible matrix within $\mathbb{R}^{d\times d}$. Without loss of generality, the activation function σ is not necessarily invertible but can be regarded as a truncation or a compressed function on the linearly transformed sample $\bar{s}W$ with finite precision. For simplicity, we assume $T(\bar{s})$ is a d'-dimensional vector within a bounded discrete set $\mathcal{K}$, and we use $|\mathcal{K}|$ in the following to denote the total number of elements in $\mathcal{K}$. As for the random weight W, we assume W is selected to be an inverse of a *β-uniform matrix*. Here, a β-uniform matrix denotes a random matrix whose elements are i.i.d. randomly selected from $[-\beta, \beta]$, denoted by $U[-\beta, \beta]$.

We first present a high-level picture of the subsequent analysis on the lower bound of sample complexity. With a similar reasoning as shown in Section 3.2, the process of data generation and inference forms a Markov Chain, $\bar{s} \to \bar{s}W \to \sigma(\bar{s}W) \to Adv$, for any estimator of the inverse of T, which can be regarded as a composition of W^{-1} and σ^{-1}. [15] Thus, $I(W^{-1}, \sigma^{-1}; Adv) \leq I(W^{-1}, \sigma^{-1}; (s_a, T(s_a + s_p)))^m)$ which implies

$$m \geq \frac{I(W^{-1}, \sigma^{-1}; Adv)}{I(W^{-1}, \sigma^{-1}; s_a, T(s_a + s_p))}.$$

[14] Gaussian input assumptions on training data are commonly used in existing works [CSZ20, DKS17, DKKZ20].

[15] With a slight abuse of notations, when the activation function σ is not invertible, we take σ^{-1} as an oracle such that $\sigma^{-1}(\sigma(s)) = s$.

With a closer look at the numerator, we have $I(W^{-1}, \sigma^{-1}; Adv) = I(\sigma^{-1}; Adv) + I(W^{-1}; Adv|\sigma^{-1}) \geq I(W^{-1}; Adv|\sigma^{-1})$, where instead we may shift our focus to the approximation error of W^{-1}. The following lower bound of $I(W^{-1}; Adv|\sigma^{-1})$ involves an application of Fano's inequality. Before proceeding, we first introduce the notion of an (ϵ, δ) packing set.

Definition 7 (**(ϵ, δ) packing set of $\mathcal{C}$**). *For z following a distribution P and a given function set $\mathcal{C}$, an element subset $\{c_1, c_2, ..., c_n\}$ of $\mathcal{C}$ is called an (ϵ, δ) packing set if for any $i \neq j$,*

$$\Pr_z(\|c_i(z) - c_j(z)\| \geq \epsilon) \geq \delta.$$

Now, in the context where $z = \bar{s}W$ and $\mathcal{C}$ is $[-\beta, \beta]^{d\times d}$, the domain of W^{-1}, we have the following lemma w.r.t. the approximation to W^{-1}.

Lemma 4. *For any given W and an arbitrary $(2\epsilon, 2\delta)$ packing set in the support domain of W^{-1}, there exists at most one element $\tilde{W}^{-1}$ within the packing set such that*

$$\Pr(\|(\bar{s}W)\tilde{W}^{-1} - \bar{s}\| < \epsilon) \geq 1 - \delta.$$

Proof. Suppose there exist two elements W_1^{-1} and W_2^{-1} within the $(2\epsilon, 2\delta)$ packing set both satisfying (21): $\Pr(\|(\bar{s}W)W_1^{-1} - \bar{s}\| < \epsilon) \geq 1 - \delta$ and $\Pr(\|(\bar{s}W)W_2^{-1} - \bar{s}\| < \epsilon) \geq 1 - \delta$. On the other hand, $\|(\bar{s}W)W_1^{-1} - (\bar{s}W)W_2^{-1}\| \leq \|(\bar{s}W)W_1^{-1} - \bar{s}\| + \|(\bar{s}W)W_2^{-1} - \bar{s}\|$. Therefore, we have

$$\Pr(\|(\bar{s}W)W_1^{-1} - \bar{s}\| < \epsilon \wedge \|(\bar{s}W)W_2^{-1} - \bar{s}\| < \epsilon) \leq \Pr(\|(\bar{s}W)W_1^{-1} - (\bar{s}W)W_2^{-1}\| < 2\epsilon).$$

On one hand, with a union bound,

$$\begin{aligned}\Pr\left(\|(\bar{s}W)W_1^{-1} - \bar{s}\| < \epsilon \wedge \|(\bar{s}W)W_2^{-1} - \bar{s}\| < \epsilon\right) \\ \geq \Pr(\|(\bar{s}W)W_1^{-1} - \bar{s}\| < \epsilon) + \Pr(\|(\bar{s}W)W_2^{-1} - \bar{s}\| < \epsilon) - 1 \geq 1 - 2\delta.\end{aligned} \tag{22}$$

On the other hand, with Definition 7,

$$\Pr(\|(\bar{s}W)W_1^{-1} - (\bar{s}W)W_2^{-1}\| < 2\epsilon) < 1 - 2\delta,$$

which is a contradiction.

It remains to use the above packing set notion to construct a covering over the support set of W^{-1}. Then, we can generalize Fano's inequality to derive a lower bound for the numerator. The conclusion is stated as follows.

Theorem 3. *Let T be a one-layer random neural network with input data and a weight matrix W that is an inverse of a $d \times d$ β-uniform matrix, as assumed above. Given an arbitrary estimation mechanism Adv satisfying (ϵ, δ) PAC approximation* (21), then

$$I(W^{-1}, \sigma^{-1}; Adv) \geq d^2 \log(\beta/\Delta) - 1$$

where ϵ and δ satisfy the following, $\delta = (1 - \Phi(2\epsilon c/\sqrt{\tau}\Delta))(1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2})$. Here, $\tau = \tau_a + \tau_p$, and Δ and c are free parameters. $\phi(t)$ is the cumulative distribution function of the normal distribution $N(0, 1)$, i.e., $\Phi(t) = 1/\sqrt{2\pi}\int_{-\infty}^{t} e^{-t^2/2}dt$.

The proof of Theorem 3 is in Appendix E. As mentioned before, $I(W^{-1}, \sigma^{-1}; Adv)$ measures how much information is required to support an efficient (ϵ, δ)-PAC approximation. Theorem 3 matches our intuition: the lower bound gets larger with a bigger τ, i.e., input data of higher entropy (since Δ can be smaller for a larger τ to achieve the same δ), or a bigger β, which corresponds to a larger concept size of transformation. In the following theorem, we provide an upper bound on the denominator $I(W^{-1}, \sigma^{-1}; s_a, T(s_a + s_p))$.

Theorem 4. *Under the same setup on the W, σ and input data,*

$$I(W^{-1}, \sigma^{-1}; s_a, T(s_a + s_p)) \leq \log |\mathcal{K}| + \frac{d}{2} \log(\frac{\tau_a + \tau_p}{\tau_p}).$$

The proof of Theorem 4 is included in Appendix F. Putting Theorems 3 and 4 together, we have the following PAC approximation resistance guarantee for Algorithm 3:

Corollary 2. *With the above setup, when the number of samples*

$$m \leq \frac{d^2 \log(\beta/\Delta) - 1}{\log |\mathcal{K}| + \frac{d}{2} \log(\frac{\tau_a + \tau_p}{\tau_p})}$$

Algorithm 3 is resistant to arbitrary (ϵ, δ)-PAC approximation, where ϵ and δ satisfy the following, $\delta = (1 - \Phi(2\epsilon c/\sqrt{\tau}\Delta))(1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2})$, with $\tau = \tau_a + \tau_p$, and free parameters Δ and c.

PAC Security of Transformation In this section, we will set the privacy concern as whether an adversary can approximate the transformation itself well. In this case, we will not impose any restrictions on the activation function. Without loss of generality, in the transformation $\sigma(\bar{s}W)$, we simply assume σ to be the identity or equivalently any invertible function and the random weight $W \in \mathbb{R}^{d \times d'}$. We call a mechanism $\mathcal{M}$ based on the observation $(s_a, T(\bar{s}))^m$ an (ϵ, δ) PAC-approximation to the random weight W used [16] in transformation T if

$$\Pr_{\bar{s}, T}(\|\mathcal{M}(\bar{s}) - \bar{s}W\| < \epsilon) \geq 1 - \delta. \tag{23}$$

Here, we also provide another random weight selection of W, where we assume each column of W, i.e., $W(:, i), i = 1, 2, ..., d'$, follows a distribution such that $\|W(:, i)\|$ is independently and uniformly distributed within an interval $[1, \beta + 1]$. With a similar reasoning as Theorem 3 and Theorem 4, we have the following theorem whose proof is in Appendix G.

Theorem 5. *Under the above assumptions, when*

$$m \leq \frac{d' \log(\beta/\Delta) - 1}{d'[\Psi(\beta, \tau_a + \tau_p)) - (\frac{\log(2\pi e \tau_p)}{2} + \frac{\beta+1}{\beta} \log(\beta + 1) - 1)]},$$

there does not exist any (ϵ, δ)-PAC approximation to the random weight W of transformation T. Here, ϵ and δ satisfy

$$\delta = \frac{1}{2}(1 - d'(2\Phi(\frac{2\sqrt{2}\epsilon}{\sqrt{d'(\tau_a + \tau_p)}\Delta}) - 1)),$$

[16] Since we do not put any specific assumptions on the activation function σ, we stick to the recovery of the random weight W in the following. But with some proper assumptions on the continuity, one can generalize the results to the approximation to $T = \sigma \circ W$.

and

$$\Psi(\beta,\tau) = -\int_0^\infty \frac{1}{\sqrt{2\pi\tau}\beta}\big(\Gamma(0,\frac{a^2}{2\tau(\beta+1)^2}) - \theta\big)\log\big(\frac{1}{2\sqrt{2\pi\tau}\beta}\big(\Gamma(0,\frac{a^2}{2\tau(\beta+1)^2}) - \theta\big)\big)\, da,$$

where $\Gamma(x,y)$ is the upper incomplete gamma function that $\Gamma(x,y) = \int_y^\infty t^{x-1}e^{-t}dt$, and $\theta = \Gamma(0,\frac{a^2}{2\tau})$. Δ is a free parameter.

As a short summary, either in data privacy (Corollary 2) or model security (Theorem 5), the private sample entropy improves both the numerator and denominator of the sample complexity bound. With the assistance of private samples, the total information required for a satisfied adversarial inference gets larger while the information provided per mixed sample decreases. In Theorem 5, such an amplification factor on sample complexity appears in a form $\mathcal{O}(\log(\tau_p)\tau_p)$, when we take the remaining parameters as constants. Such a claim also holds if W is selected to be a uniform matrix. Indeed, with a uniform matrix W, similar to Corollary 2, the numerator of the sample complexity in Theorem 5 can be further improved to be $\Theta(d'd)$[17]. The reason that we assume W in the current form is just to simplify the denominator calculation. Therefore, if W is selected to be an inverse of a uniform matrix or a uniform matrix itself in Corollary 2 and Theorem 5, respectively, and we measure the power of randomness of W to resist adversarial inference (taking output quantification precision and other security parameters as constants), the sample complexity required in both cases is $\Theta(d)$, the dimensionality of the samples.

5 Experimental Results

This section consists of four main experiments, where we set out to further study the following problems: transformation of perfect secrecy, random neural net transformation and fully-connected network training, the hardness of recovery of private images, and the match between transformation and training algorithm. The *Adult* dataset from UCI, and two standard image-datasets, *MNIST* and *CIFAR-10* are the data sources for the following empirical results.

5.1 Perfect Secrecy

Experiments in this section were run in Matlab R2020a.

Ridge Regression on *Adult*: In Section 2, we theoretically study how to achieve perfect secrecy for the generalized linear model. We take the Adult dataset as the first example. The Adult consensus data contains 14 attributes including education level, age, gender and occupation. The task here is to develop a predictor that predicts whether an individual's annual income is above \$50K.

To obtain perfect secrecy, we consider transforming the original data to some independent values. To specify the operations, the input feature set $X \in \mathbb{R}^{n\times 14}$ is first transformed to a square matrix XW, with a random matrix $W \in \mathbb{R}^{14\times n}$. For regularization, we apply SVI on the SVD form of $XW = U\Sigma V$ by $\tilde{X} = U(\Sigma+\tau\cdot\boldsymbol{I}_n)V$. Then, we define the transformation on the feature domain $x \in \mathcal{X}$ as $T(x) = x\cdot W\cdot\tilde{X}^{-1}$, and fix the transformed training dataset as a constant $\boldsymbol{I}_n$. We take 4,000 samples, 2,000 from each class, as the test data. When the training number $n = 20,000$ and $n = 30,000$, the Non-Private (non-private) ridge regressions achieve 63.3% and 66.5% accuracy, respectively. As a comparison, when we select $\tau = 1$, the perfect-secrecy ridge regressions achieve 63.2% and 66.2% accuracy, respectively.

[17] Also from our experiments, the two W generation approaches have almost the same utility.

Multi-classification on *MNIST*: As mentioned before, a neural network is a special generalized linear model, and we further consider multi-class classification with perfect secrecy. We take the MNIST dataset, the images of 10 handwritten digits, as an example. We consider the following neural network architecture, which is formed by three fully-connected layers of 300, 200 and 100 neurons, respectively, which are connected by Relu activation functions, and a softmax classification layer at the end. Instead of using the full MNIST data, we conduct and compare several subclassification tasks, where the number of classes c varies from 2 to 10. Correspondingly, for given c, we take 2000 samples of each digit from 1 to c as the training dataset. Similarly, we transform MNIST data via the above-mentioned semilinear transformation with SVI of $\tau = 1$. Under the same training setup, we compare the training over original data and transformed data in Table 1.

Table 1: **Handwriting Digit Classification of *MNIST* with Perfect Secrecy via Fully-connected Network**

Methods/Accuracy	2-Classify	4-Classify	6-Classify	8-Classify	10-Classify
Non-Private Training	99.4%	98.2%	97.9%	97.2%	97.0%
Perfect Secrecy Training	98.3%	94.8%	90.1%	85.6%	82.2%

The experiments suggest that as the data and the classification task become more complicated (the number of classes gets larger), the trained network is more prone to be over-fitted. As shown in Table 1, the accuracy (generalization) of perfect secrecy training drops more sharply compared to the baseline. Recall the analysis in Section 2.3, an arbitrary neural network on an invertible linearly-transformed dataset is bijective to a same size neural net, which shares the same training error on the original data. In other words, the described perfect-secrecy training can be viewed as randomly finding a network of satisfied training error. As tasks get more complicated, those nets which generalize well are sparser amongst all nets of small training error. We believe the above data-independent training can provide an interesting perspective to study generalization in neural networks [JGH18, LL18], and may be of independent interest.

5.2 Data Mix and Random Transformation

We present experiments on the effects of data processing proposed in Algorithm 3 on training.
MNIST: These experiments were conducted using an implementation of Algorithm 3 in Matlab R2020a. The results use an Intel Core i7 CPU @3.1GHz, with 16GB of memory. We use the entire 60,000 training samples of MNIST. As for the random transformation, we use a one-layer neural network, where the number of neurons equals the dimension of samples. Therefore, after applying the data mixing over the original samples set $\mathbb{R}^{n\times d}$, we generate m mixed samples in a form $\mathbb{R}^{m\times d}$. Through the transformation, the size of transformed data does not change, which is still $\mathbb{R}^{m\times d}$. In developing the PAC based privacy proof in Section 4, we proposed two approaches to generate the weight matrix $W \in \mathbb{R}^{d\times d}$ for sample transformation:

- Approach 1: W is the inverse of a matrix, whose each entry i.i.d. follows a uniform distribution $U[-\beta, \beta]$.
- Approach 2: Each row of W is i.i.d. selected such that the l_2 norm of the vector is uniformly distributed as $U[1, 1+\beta]$.

Approach 1 and 2 correspond to the weight generations described in Corollary 2 and Theorem 5, respectively. In the following, we select $\beta = 50$ in both approaches. In Table 2, we include the details of the training time (measured as SGD iterations with a batch size of 128) and classification accuracy achieved. Here, we

select the network to be trained as a 5-layer structure with 4 fully-connected layers of 256, 128, 64, 32 neurons, respectively, and a regression layer at the end. We run 20 trials of each case listed in Table 2, where averages and standard deviations (in parentheses) are provided. The cost of each iteration is 0.022 seconds and 0.023 seconds, for the no-mix non-private and the mix (non-private and private) training, respectively. The difference is because of the slightly different empirical loss functions produced by original and mixed samples, respectively. From Table 2, it is clear that the weight matrix generated by Approach 2 produces more

Table 2: **Handwriting Digit Classification of *MNIST* with Fully-connected Neural Network**

Methods	Weight Generation	# Samples	Iterations (10^3)	Accuracy(%)
Non-Private Training	n.a.	60,000	5.8 ($\pm$0.9)	98.5 ($\pm$0.07)
2-Mix Non-Private Training	n.a.	120,000	9.2 ($\pm$1.1)	98.3 ($\pm$0.08)
2-Mix Non-Private Training	n.a.	180,000	15.1 ($\pm$0.3)	98.6 ($\pm$0.08)
2-Mix Transformed Training	Approach 1	120,000	47.8 ($\pm$13.7)	96.9 ($\pm$0.7)
2-Mix Transformed Training	Approach 1	180,000	76.2 ($\pm$29.7)	97.2 ($\pm$1.8)
4-Mix Transformed Training	Approach 1	120,000	48.3 ($\pm$7.8)	95.4 ($\pm$2.0)
4-Mix Transformed Training	Approach 1	180,000	68.9 ($\pm$18.4)	95.7 ($\pm$2.4)
2-Mix Transformed Training	Approach 2	120,000	15.6 ($\pm$1.7)	98.3 ($\pm$0.07)
2-Mix Transformed Training	Approach 2	180,000	19.7 ($\pm$2.0)	98.4 ($\pm$0.10)
4-Mix Transformed Training	Approach 2	120,000	17.9 ($\pm$1.5)	97.8 ($\pm$0.13)
4-Mix Transformed Training	Approach 2	180,000	19.3 ($\pm$2.2)	98.0 ($\pm$0.08)

stable and efficient learning over transformed data. One key factor which influences both the training time and performance of transformed data is the distribution of singular values of the weight matrix used in the transformation network. Intuitively, a weight matrix in which the largest and smallest singular values differ by a lot will make the transformed data more fuzzy. In Approach 1, with certain probability, a uniform matrix can be quite close to singular and the largest singular value of the matrix inverse is much larger than that of the matrix itself. The probability with which this occurs becomes higher when the dimensionality increases resulting in more unstable learning performance. However, a straightforward tradeoff between the utility and privacy is rejecting *outlier* random matrices generated prior to data transformation and exposure, and the consequent security loss can be modelled by an additional failure probability.[18]

CIFAR-10: Table 3 provides results on CIFAR-10 using the same network we specified above with an implementation of Algorithm 3 in PyTorch 1.7.1. The results use an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, with 32GB of memory, and a GeForce GTX 1060 GPU with 6GB of memory. Each run was 30 epochs with a batch size of 128, and the cost of each iteration is 0.003897 seconds and 0.004033 seconds, for the no-mix non-private and the mix (non-private and private) training, respectively, on the GPU. Client-side time for mixing and transformation generation and implementation on the CPU was less than 250 seconds for 400,000 mixed samples.

Given our use of a relatively simple fully connected network, the Non-Private training accuracy and the Mix Non-Private training accuracy are around 50%. Accuracy numbers are averaged across 3 trials. Mix Transformed data training accuracy is close to the Non-Private mix training for both approaches. We used the

[18] It is worthwhile mentioning that Approach 1 and Approach 2 shown in Corollary 2 and Theorem 5 produce different and incomparable PAC security guarantees, where the objectives of concern are data privacy and transformation security, respectively.

outlier rejection approach mentioned above for Approach 1. We reject the random matrices generated if the ratio between the largest and the smallest singular values is $\geq 1.15 \times 10^4$.

Table 3: **Object Detection of *CIFAR-10* with Fully-connected Neural Network**

Methods	Weight Generation	# Samples	Iterations (10^3)	Accuracy (%)
Non-Private Training	n.a.	60,000	14.0	55.59
2-Mix Non-Private Training	n.a.	100,000	23.4	50.35
2-Mix Non-Private Training	n.a.	200,000	46.8	51.97
2-Mix Non-Private Training	n.a.	300,000	70.3	53.12
2-Mix Non-Private Training	n.a.	400,000	93.7	53.67
2-Mix Transformed Training	Approach 1	100,000	23.4	40.30
2-Mix Transformed Training	Approach 1	200,000	46.8	46.49
2-Mix Transformed Training	Approach 1	300,000	70.3	48.98
2-Mix Transformed Training	Approach 1	400,000	93.7	48.67
2-Mix Transformed Training	Approach 2	100,000	23.4	49.58
2-Mix Transformed Training	Approach 2	200,000	46.8	52.64
2-Mix Transformed Training	Approach 2	300,000	70.3	52.96
2-Mix Transformed Training	Approach 2	400,000	93.7	53.41

5.3 Transformed Data Recovery

We conducted experiments to measure the hardness of inverting transformed data. Depending on specific prior knowledge, attacks may also vary. In the following experiments, we aim to model an attack where an adversary only has full knowledge on the public data and correspondence. The results provide some intuition on the relationship between transformation, data mixing and privacy.

To relax restrictions on the adversary, we assume that the adversary knows all the underlying correspondences and the random weight for those public samples used in the mixing and transformation. This is also the assumption made in Corollary 2 and Theorem 5. To give an example, suppose samples from two subsets $\hat{S}_p \in S_p$ and $\hat{S}_a \in S_a$ within private S_p and public S_a sample sets, respectively, are mixed in a form $\sum_{s_i \in \hat{S}_p} \lambda_i s_i + \sum_{s_j \in \hat{S}_a} \lambda_j s_j$. The adversary will know the corresponding $T(\sum_{s_i \in \hat{S}_p} \lambda_i s_i + \sum_{s_j \in \hat{S}_a} \lambda_j s_j)$ and public sample(s) $\hat{S}_a$ and λ_j involved. We assume the adversary takes those "noisy" samples with $\sum_{s_j \in \hat{S}_a} \lambda_j s_j$ as the label and the observation $T(\sum_{s_i \in \hat{S}_p} \lambda_i s_i + \sum_{s_j \in \hat{S}_a} \lambda_j s_j)$ as the feature, and trains a network to mimic the inversion of transformation T.

Details of experiments can be found in Appendix A; we summarize the main observations here:

- Recovery is possible in MNIST if enough mix samples (and their correspondences) are provided. Recovery is more difficult given fewer samples. There is a budget associated with the number of samples that can be exposed to inhibit recovery, and we provided a theoretical basis for this budget.
- Recovery is more difficult for more complex datasets such as CIFAR-10 in comparison with simpler datasets like MNIST. Our theory explains this; compared to CIFAR-10, MNIST is much sparser and under our setup it corresponds to a smaller entropy of the private data.
- Data mixing produces a big security amplification: under the same setup, data recovery is much easier (a smaller sample complexity required) without data mixing, becoming a simple chosen plaintext attack.

5.4 Matching the Transformation to the Training Algorithm

As the first key challenge described in Section 1.1, a proper transformation is such that the transformed data matches the subsequent training algorithm. From our observation, a random fully-connected data transformation network matches well with a fully-connected or transformer neural network. The Convolutional Neural Network (CNN) provides an interesting example here. It is well-known that by applying AlexNet [KSH17] or Residual neural network (Resnet) [HZRS16], one can achieve 90+% detection accuracy on CIFAR-10. Training with a fully connected network is significantly less accurate (cf. Table 3).[19] However, if we let the image data (reshaped into a vector form) pass through a random fully-connected layer transform as before, and then conduct training with a CNN, it is no better than training original CIFAR-10 images with a fully-connected network. In images, neighboring pixels are potentially more correlated compared to faraway ones, and thus the feature maps generated by the convolutional kernel are more efficient in extracting important information. One can therefore hypothesize that an image transformed by a fully-connected layer will lose locality, since each coordinate in the transformed image is an aggregation of all pixels from the original image. This can be viewed as a feature map where the kernel size equals that of the image itself. Therefore, the first convolutional layer is not a neighborhood operation on such transformed samples. Thus, for such transformed data, it may be just as or more efficient to use a fully-connected network rather than a CNN. To match a CNN, a more-friendly transformation could be a random convolutional layer. In our preliminary experiments with random convolutional layer transforms, the accuracy gap compared to non-private CNN training over CIFAR-10 images is within a few percents. However, our privacy proof cannot directly be generalized to the random convolution case, and therefore we leave a comprehensive study to future work.

6 Private Model Release and Model Extraction Defense

With the above preparation, we now return to the challenges we pointed out in Section 1, i.e., a unified framework to accommodate both *private learning*, outsourced to a server, and *private inference*, controlling both data privacy leakage as well as the privacy of the user's model. Recent studies have shown that even with black-box accesses, one can recover sensitive information or even the full machine learning model [FJR15],[TZJ+16]. Such inversion can heavily compromise the privacy of data, which the model is trained over, and the intellectual property of the model holder.

For the former challenge, i.e., the privacy leakage of training data during the post-processing of the private model, if the training procedure is differentially private, for example an SGD protocol perturbed by Laplace or Gaussian Mechanism [ACG+16], there will not be any problems since DP is resistant to postprocessing. However, as we mentioned in Section 1, almost all existing DP works assume that the computation has to be implemented by the user. To resort to untrusted cloud service, an LDP mechanism on the original data seems to be the only approach. On the other hand, even if the computing process is protected through a cryptographic approach, such as homomorphic encryption, the sensitivity calculation in general is NP-hard [XT08] and thus it is hard to apply a DP mechanism for the postprocessing operations. In the following, we describe how to quantify such leakage when applying the model for inference.

Data Privacy Protection: Algorithm 4, an inference protocol over mixed samples, is inspired by [LWG+20] which shows that mixup can not only improve the performance of training but also inference. This coincides with our intuition that, taking $0.5(x + x')$ as input, a well-trained predictor should output the mixed label of $0.5(x + x')$. We may consider the worst case that the query client and server are colluding and thus f_T is also known to the adversary. In Algorithm 4, y', the private label, is unknown to the adversary. Also in

[19] From a training perspective, when we feed images to a fully-connected network directly, the number of parameters to be optimized is large, since image dimension is usually large, thereby making training difficult.

Algorithm 4 Private Application of a c-classification Model for Inference

Input: A private dataset $\mathcal{D}$; Label bases of c classes $b_1, b_2, ..., b_c$ and Transformation T generated by user; Model f_T trained over transformed dataset with T by server and sent back to user;

1: A query with a feature x is sent to the user by some query client;
2: User randomly selects a sample (x', y') from $\mathcal{D}$ and produces a mixed feature $0.5(x + x')$.
3: Apply transformation T over $0.5(x + x')$ and calculate $f_T(T(0.5(x + x')))$;
4: Approximate the true label of x by $2[f_T(T(0.5(x + x'))) - 0.5y']$ and compute the nearest label basis:

$$i^* = \arg\min_{i\in[1:c]} \|2[f_T(T(0.5(x + x'))) - 0.5y'] - b_i\|.$$

Output: i^*-th class as the inference result.

practice, most commonly used predictors, such as a neural network with non-linear activation functions, are not invertible. To relax the restrictions, we still assume f_T is invertible and that the adversary can even recover $T(x + x')$. Therefore, provided the query feature x and $T(x + x')$, the privacy loss caused during inference is reduced to the PAC security model described in Section 4 when analyzing Algorithm 3. Hence, through Algorithms 3 and 4, the PAC security model provides a unified analysis for privacy leakage during both learning and inference over transformed data. Applying Corollary 2 which connects the privacy budget (ϵ, δ) to the sample complexity, when we take the entire data privacy loss during model training and model application into account, the sample complexity m should cover both the training samples exposed and the later inference queries.

Model Extraction Defense: The second challenge, i.e., one may steal the model or approximate $f_T \circ T$ through multiple accesses, is indeed also resolved by Algorithm 4. Such approximation hardness is quantified in Theorem 5, where we measure the sample complexity to find a mapping which is statistically close to T. Thus, even if the query client and the server are colluding where f_T is known, adversaries cannot steal the model without efficient approximation of T. Similarly, provided (ϵ, δ) security budget of the model itself, the sum of the numbers of exposed training samples and the later inference queries should be no bigger than the sample complexity m.

Putting things together, a unified private learning and inference framework is constituted through a combination of Algorithms 3 and 4. The proposed protocols with PAC security represent a framework where the privacy/security losses in four dimensions, namely, the data privacy and the model security loss [CJM20], during the training and the model application (inference) are uniformly measured [20].

7 Related Works

Data Mixing and Privacy: A popular data augmentation method *mixup* was proposed in [ZCDLP18] and has been widely studied both empirically [BCG$^+$19, TCB$^+$19, VLB$^+$19, PXZ19], and theoretically [ZDK$^+$20]. *Mixup* achieves significant success in semi-supervised learning [BCG$^+$19] and has been shown to strengthen the robustness even under adversarial attacks [PXZ19]. We are not the first to consider introducing *mixup* for privacy preservation. Huang et al. proposed a private training protocol, Instahide in [HSLA20], and Liu et al. proposed a private inference protocol, Datamix in [LWG$^+$20]. Instahide performs sample-specific sign-flipping unlike the uniform transformation over each sample in the Dauntless framework. Straightforward applications of *mixup* may encounter two major vulnerabilities from adversarial inference.

[20] If one wants to achieve the PAC security w.r.t. both data, with budget (ϵ_1, δ_1), and the model, with budget (ϵ_2, δ_2), simultaneously, the sample budget should be the minimum of that determined by their respective security budgets.

Recall the description of *mixup* in Section 3.2. First, the trivial one-hot label encoding exposes both the random weights used for mixing. Second, though data can be randomly mixed, the underlying permutation cannot provide rigorous privacy guarantees. [21] Carlini et al. [CDG+20] approximates the sample correspondence in Instahide with a similarity graph, and follow-up works [CSZ20, HST+20], which are based on the phase retrieval model, present several attacks on Instahide.

Transfer Learning and Data Transformation: Generally speaking, transfer learning studies how the experience or knowledge gained in solving one model could be reused for other tasks. Transfer learning has been well studied recently [PY09, TS10, ZQD+20]. Especially in deep learning, it is common to use a pre-trained network for a large and challenging classification task as a basis, over which the new model is further trained. The framework of perfect secrecy training shown in Section 2 can also be presented in a transfer learning view: for privacy preservation, we map private data to some public data and then incorporate the transformation and the corresponding model of the public data. In Section 2, we only studied a linear transformation and set the transformed data to be constant (identity). We leave more comprehensive investigation on the transformation, which may be more suitable to transfer learning, to future work.

8 Conclusion

Exploiting the fundamental statistical nature of machine learning, we have proposed a unified private learning and inference framework that implements a training procedure on a transformed sample domain to produce a usable model. We propose a new information theoretic security metric, namely PAC inference resistance. Different from the classic view where security/privacy loss is measured by the additional information provided by ciphertext, PAC security further takes the prior knowledge into account providing a more comprehensive characterization.

Our preliminary experiments in the Dauntless framework show promise in producing usable models from transformed datasets, in a computationally-efficient manner. The proposed framework sheds light on a possibly unified analysis on data privacy, model security, data feature extraction and training from an information processing perspective. Advances in machine learning with more efficient feature extraction and training algorithms may enhance data protection, by allowing more flexible designs of sample transformation that match the training. Conversely, studies on sample transformation and security in an information processing view may help advance the understanding of machine learning mechanisms.

[21] A hypothesis in [HSLA20] is that the computational hardness of *mixup* can be reduced to the subset sum problem.

Bibliography

[AAUC18] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018.

[AC11] Jean-Yves Audibert and Olivier Catoni. Robust linear least squares regression. *The Annals of Statistics*, 39(5):2766–2794, 2011.

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[AD96] Javed A Aslam and Scott E Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters*, 57(4):189–195, 1996.

[AHW+96] Peter Auer, Mark Herbster, Manfred K Warmuth, et al. Exponentially many local minima for single neurons. *Advances in neural information processing systems*, pages 316–322, 1996.

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 410–419. IEEE, 2012.

[BCG+19] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[Bha07] Rajendra Bhatia. *Perturbation bounds for matrix eigenvalues*. SIAM, 2007.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[CDG+20] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. An attack on instahide: Is private learning possible with instance encoding? *arXiv preprint arXiv:2011.05315*, 2020.

[CJM20] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *Annual International Cryptology Conference*, pages 189–218. Springer, 2020.

[CSS12] Kamalika Chaudhuri, Anand Sarwate, and Kaushik Sinha. Near-optimal differentially private principal components. In *Advances in Neural Information Processing Systems*, pages 989–997, 2012.

[CSZ20] Sitan Chen, Zhao Song, and Danyang Zhuo. On instahide, phase retrieval, and sparse matrix factorization. *arXiv preprint arXiv:2011.11181*, 2020.

[DJW13] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 2013.

[DKKZ20] 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*, pages 1514–1539. PMLR, 2020.

[DKS17] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–84. IEEE, 2017.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[DRS19] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

[DRV10] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[DTTZ14] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2014.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[GH01] Claudio Gentile and David P Helmbold. Improved lower bounds for learning from noisy examples: an information-theoretic approach. *Information and Computation*, 166(2):133–155, 2001.

[HSLA20] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning*, pages 4507–4518. PMLR, 2020.

[HST+20] Baihe Huang, Zhao Song, Runzhou Tao, Ruizhe Zhang, and Danyang Zhuo. Instahide's sample complexity when mixing two private images. *arXiv preprint arXiv:2011.11877*, 2020.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[KKMM13] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. Privacy via the johnson-lindenstrauss transform. *Journal of Privacy and Confidentiality*, 5(1):39–71, 2013.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[LL18] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[LWG+20] Zhijian Liu, Zhanghao Wu, Chuang Gan, Ligeng Zhu, and Song Han. Datamix: Efficient privacy-preserving edge-cloud inference. In *Computer Vision – ECCV 2020*, pages 578–595. Springer International Publishing, 2020.

[Mir17] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[MSM17] Paulo Martins, Leonel Sousa, and Artur Mariano. A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–33, 2017.

[NW72] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[PCS+20] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. *ICLR 2020 Submission (openreview.net)*, 2020.

[PXZ19] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. *arXiv preprint arXiv:1909.11515*, 2019.

[PY09] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[TCB+19] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32:13888–13899, 2019.

[TS10] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[TZJ+16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, 2016.

[VLB+19] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[WGC19] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019.

[WLF16] Yu-Xiang Wang, Jing Lei, and Stephen E Fienberg. On-average kl-privacy and its equivalence to generalization for max-entropy mechanisms. In *International Conference on Privacy in Statistical Databases*, pages 121–134. Springer, 2016.

[XT08] Xiaokui Xiao and Yufei Tao. Output perturbation with query relaxation. *Proceedings of the VLDB Endowment*, 1(1):857–869, 2008.

[ZCDLP18] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[ZDK+20] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

[ZLX+20] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 493–506, 2020.

[ZQD+20] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[ZZ18] Lijun Zhang and Zhi-Hua Zhou. l_1 regression with heavy-tailed distributions. In *Advances in Neural Information Processing Systems*, pages 1076–1086, 2018.

A Recovery of Private Images from Transformed Data

We first consider the following example, where in the MNIST dataset with 60,000 training samples, all the samples of digits from 1 to 5 are private while the rest 6 to 0 are all public. In this case, S_p corresponds to all samples of digits 1-5 and S_a corresponds to the rest. As mentioned in Section 5.3, we assume the adversary takes those "noisy" samples with $\sum_{s_j \in \hat{S}_a} \lambda_j s_j$ as the label and the observation $T(\sum_{s_i \in \hat{S}_p} \lambda_i s_i + \sum_{s_j \in \hat{S}_a} \lambda_j s_j)$ as the feature, and trains a network to mimic the inversion of transformation T. Unless otherwise stated, we used Approach 1 (cf. Section 5.2) to generate the transformed data.

In our first experiment, we consider the scenario where 180,000 mixed samples are randomly generated as described above, where we randomly select 10 private mixed images with their corresponding transformed ones and the recovery is shown in Fig. 2. In Fig. 2, the first row corresponds to the private part of selected images, the second row corresponds to the transformed images, and the recovered images are shown in the third row. In the following experiments, since the transformed images all look like random noise, we will omit them. Clearly, as we restrict the adversary to the samples of only digits $\{6, 7, 8, 9, 0\}$, the learning with noisy data produces a predictor which applies the knowledge from those 5 classes in the public samples to approximate the unknown private data, and the recovery essentially fails. However, in practice, it is more likely that the private and public sets share samples from a same class. It is more interesting to assume that the adversary has much more prior knowledge on samples in each class.

In the second example, we change the assumption of private and public sets of MNIST in that we assume the samples of each class are evenly divided into two parts, which are included in public and private sets, respectively. In other words, the adversary knows 50% samples in each class. We still assume the correspondence and mixing weights for public samples are known by the adversary. Fig. 3 shows the recovery of 10 private pictures randomly picked. Clearly, the recovery is much more successful, where about 70% of private pictures are well-approximated. In expectation, the "noise" is almost in the same scale in the first and the second examples, as one half of MNIST pictures are unknown. However, MNIST handwriting pictures are relatively simple, where the data matrix is sparse. Provided 180,000 observations, an adversary with prior knowledge on one half of samples in each class can successfully invert the transformation T.

We now consider weakening the prior knowledge the adversary has. In Fig. 4, under the same setup, we assume only 25% of samples in each class are known by the adversary. This impedes recovery significantly. Similarly, on another dimension, we can also decrease the number of mixed samples generated. In Figs. 5 - 6, we still assume 50% samples are public but only 60,000 2-mixed samples are generated, over which the training achieves 94.9% accuracy. Under the same setup, the private pictures recovered are shown in Fig. 5. Furthermore, if we do 4-mix where the mixed images become more fuzzy, the recovery becomes more difficult. Under the same setup, training over 60,000 4-mix samples achieves 93.7% accuracy, where the recovered pictures are shown in Fig. 6.

Importantly, more complicated pictures have a potential for larger entropy. If the data to be mixed and transformed is CIFAR-10 pictures, even with prior knowledge on one half of 50,000 CIFAR-10 training samples and 200,000 mixed samples produced, the inversion becomes much harder, as shown in Fig. 7.

Finally, we show that if we simply apply a uniform transformation (Approach 1) on the original samples *without* data mixing, the attack becomes easier to carry out. In Figs. 8 and 9, under the same setup, we assume the adversaries know 50% and 95% of the CIFAR-10 training dataset, respectively, and the recovered pictures are shown. Clearly, though perfect reconstruction is impossible, a much better private image approximation is produced.

Fig. 2: Recovery of Transformed MNIST Pictures with Prior Knowledge on All Samples of Digits 6-0 and 180,000 Mixed Samples

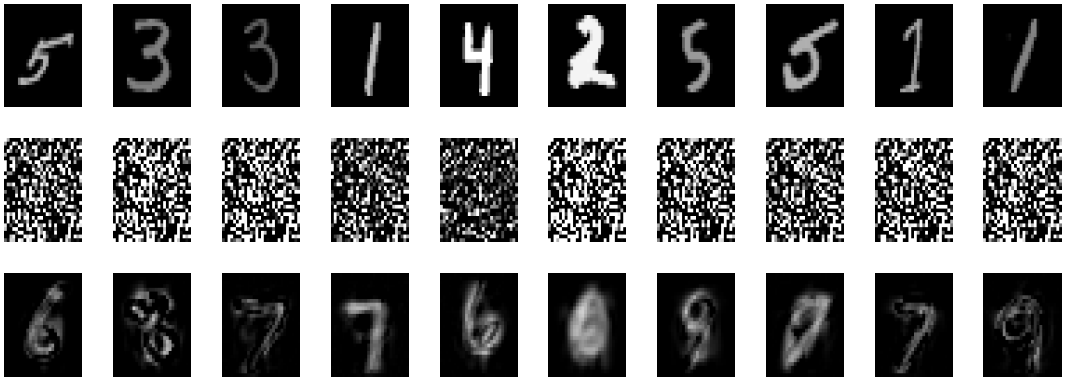


Fig. 3: Recovery of 2-Mix Transformed MNIST Pictures with Prior Knowledge on 50% of Full Dataset and 180,000 Mixed Samples

Fig. 4: Recovery of 2-Mix Transformed MNIST Pictures with Prior Knowledge on 25% of Full Dataset and 180,000 Mixed Samples

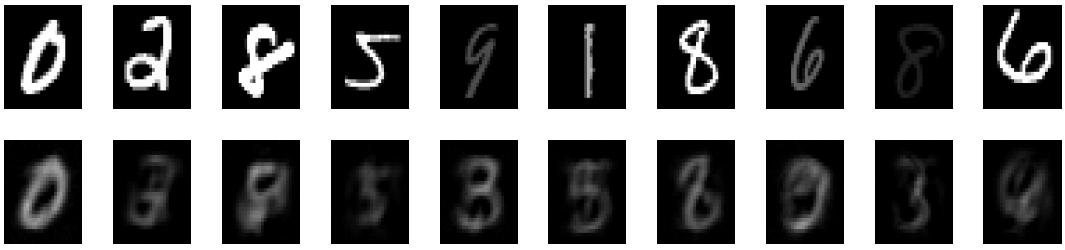


Fig. 5: Recovery of 2-Mix Transformed MNIST with Prior Knowledge on 50% of Full Dataset and 60,000 Mixed Samples

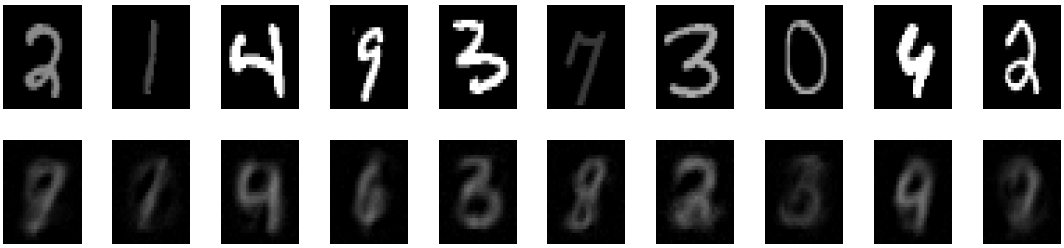


Fig. 6: Recovery of 4-Mix Transformed MNIST Pictures with Prior Knowledge on 50% of Full Dataset and 60,000 Mixed Samples

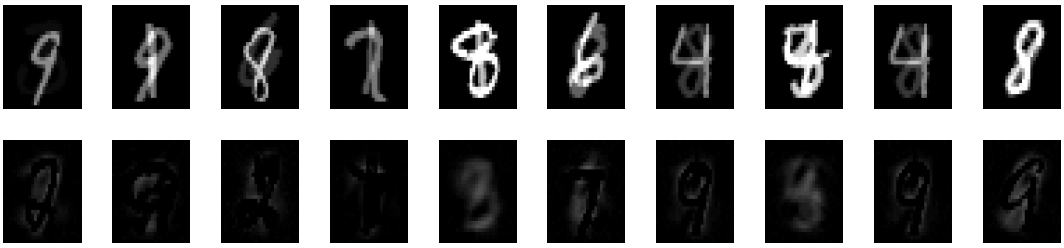


Fig. 7: Recovery of 2-Mix Transformed CIFAR-10 Pictures with Prior Knowledge on 50% of Full Dataset and 200,000 Mixed Samples

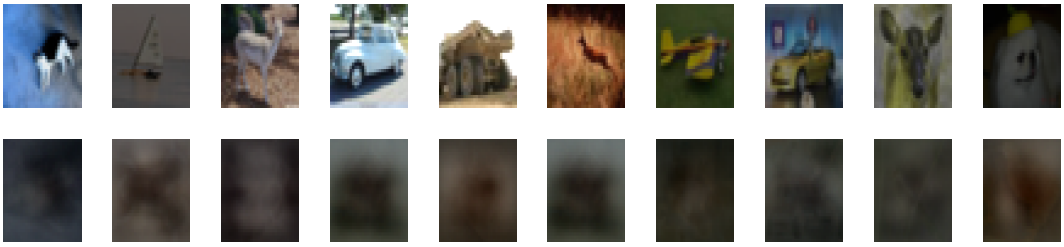


Fig. 8: Recovery of NO-Mix Transformed CIFAR-10 Pictures with Prior Knowledge on 50% of Full Dataset of 50,000 Samples

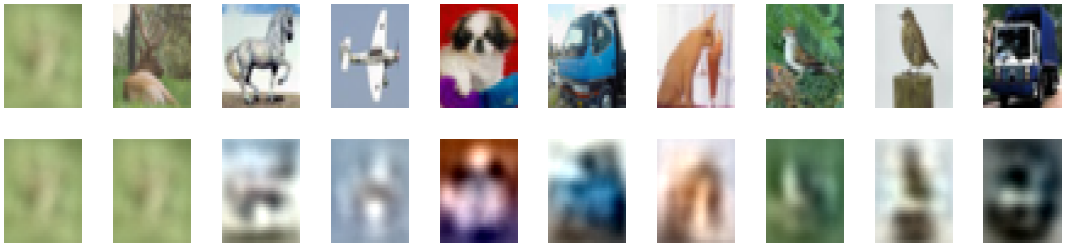
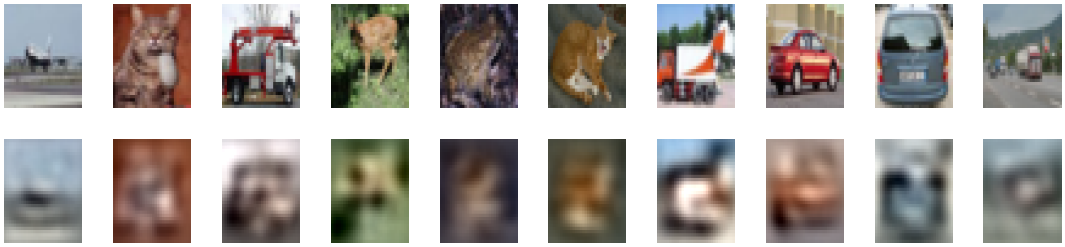


Fig. 9: Recovery of NO-Mix Transformed CIFAR-10 Pictures with Prior Knowledge on 95% of Full Dataset of 50,000 Samples



B A Transformation Construction with Ambiguity Set Comprising the Column Space

From a view of linear space, each column of XA is a linear combination of the columns of X. In other words, XA is within the linear space spanned by the columns of X. Therefore, if we restrict the transformation to be linear, in an *additional information* measurement, the least privacy loss is to only expose the column space of X. In the following, we construct a linear transformation framework such that its ambiguity set equals the column space of its input.

For simplicity, we assume X is of full rank. We divide our analysis into two cases: if $n \leq d$, then there always exists a matrix X_R^{-1} such that $XX_R^{-1} = [\boldsymbol{I}_n, \mathbf{0}_{(d-n)\times n}]$. Such a transform is of perfect secrecy w.r.t. all full rank matrices where $n \leq d$. Second, if $n > d$, clearly we cannot find such an X_R^{-1} but similarly, we aim to find some *unique feature* only determined by the column space of X. Since there are infinite d-dimensional subspaces in $\mathbb{R}^n$, resorting to a look-up table becomes impossible. To efficiently encode and construct a mapping, which is uniquely determined by a d-dimensional subspace, we provide the following scheme shown in Algorithm 5.

To justify that Algorithm 5 preserves perfect ambiguity w.r.t. all matrices enjoying the same column space as X, we need the following lemma:

Lemma 5. *For any two sets of normalized orthogonal bases $X_1 = [x_1^1, x_2^1, ..., x_d^1]$ and $X_2 = [x_1^2, x_2^2, ..., x_d^2]$ of a d-dimensional linear space, there exists a unitary matrix $U \in \mathbb{R}^d$ such that $X_1 = X_2U$.*

Algorithm 5 Perfect Column Space Ambiguity Transformation

1. Apply Gram–Schmidt Orthonormalization on the columns of $X = [x_1, x_2, ..., x_d]$ with normalization to $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_d]$.
2. Randomly generate a unitary matrix $U \in \mathbb{R}^d$.
3. Output $\hat{X} = \tilde{X}U$.
4. Determine the transform matrix $A = (X^T X)^{-1} X\hat{X}$.

Proof. Since X_1 and X_2 are two normalized orthogonal bases of one linear space, let $x_i^1 = \sum_{j=1}^d c_{ij} x_j^2$, where in a matrix form, it is $X_1 = X_2 C$. Since $\|x_i^1\| = 1$, we have $\sum_{j,l=1}^n c_{ij} c_{il} (x_j^2)^T \cdot x_l^2 = \sum_{j=1}^n c_{ij}^2 = 1$, i.e., each column of C is normalized. Furthermore, since $(x_j^1)^T \cdot x_l^1 = 0$, which implies $\sum_{i=1}^d c_{ij} c_{il} = 0$, i.e., the inner product of any two columns equals 0, which are orthogonal. Thus, C is an unitary matrix.

With the above lemma, following the orthonormalization and normalization across the columns of X, we find a set of normalized orthogonal bases of the column-spanned space. Through a random unitary matrix, any normalized orthogonal basis has identical probability to be the output. Therefore, it is indistinguishable for all elements in the same column space.

C Multiplicative Differential Privacy for Diagonal Matrix

In this section, we set out to study whether analogous to the additive DP mechanism, through selections of the multiplicative operator from a more heavy-tailed distribution, we can derive arbitrarily strong DP guarantees (arbitrarily small distinguishability advantage). Though in general, without further assumptions the answer is negative but in a special case if the input matrix is diagonal, we have the following observations.

We start with one-dimensional reals. For two real numbers r_1 and r_2, how close could the distributions of $x_1\alpha$ and $x_2\alpha$ be for some random α generated from certain distribution $\mathcal{Q}$? Without loss of generality, we assume both r_1 and r_2 are positive. Then, we have for any $z \in \mathbb{R}$

$$\frac{\Pr(r_1\alpha = z)}{\Pr(r_2\alpha = z)} = \frac{r_2}{r_1} \cdot \frac{\mathcal{P}(z/r_1)}{\mathcal{P}(z/r_2)}. \tag{24}$$

To make the above ratio as close as possible to 1, we consider the following distribution:

$$\mathcal{P}(z) = \begin{cases} \dfrac{\tau}{1+\tau}, & 0 \le z \le 1 \\ \dfrac{\tau}{1+\tau} z^{-(1+\tau)}, & z > 1 \end{cases} \tag{25}$$

for some $\tau > 0$. Then, we have the following lemma.

Lemma 6. *For random α distributed as (25), and arbitrary z,*

$$\mathbb{P}(r_1\alpha = z) \le \max\{r_1/r_2, r_2/r_1\}^\tau \, \mathbb{P}(r_2\alpha = z) + \tau/(1+\tau). \tag{26}$$

The proof of Lemma 6 is simply substituting the density function (25) to express the ratio. Here, $\tau/(1+\tau)$ corresponds to the probability that $\Pr(z \in [0, 1))$. As $\tau \to 0$, the two resultant output distributions move towards being identical. The above observation has a straightforward generalization to multidimensional space for positive diagonal matrices. Consider $X_1 = \mathcal{DIAG}[x_1^1, x_2^1, ..., x_n^1]$ and $X_2 = \mathcal{DIAG}[x_1^2, x_2^2, ..., x_n^2]$, where $\mathcal{DIAG}[*]$ denotes a diagonal matrix with diagonal elements $*$. If we select $W^{(1)} = \mathcal{DIAG}[\alpha_1, \alpha_2, ..., \alpha_n]$, where α_i i.i.d. as in (25), we have the following lemma.

Lemma 7. *For random diagonal matrix $W^{(1)}$ with i.i.d. diagonal elements distributed as (25), and two positive diagonal matrices X_1 and X_2, for arbitrary z,*

$$\mathbb{P}(X_1W^{(1)} = z) \leq \max\{det(X_1)/det(X_2), det(X_2)/det(X_1)\}^{\tau}\, \mathbb{P}(X_2W^{(1)} = z) + n\tau/(1+\tau). \quad (27)$$

Lemma 7 is a composition of Lemma 6 with the union bound. The above results on diagonal matrices render the random transformation construction of Algorithm 6.

Algorithm 6 Singular Value Obfuscation

Input: Matrix $X \in \mathbb{R}^{n\times n}$.
1. Apply SVD on $X = U\Sigma V$, where Σ is the singular value diagonal matrix;
2. Transformation by $X' = XV^TW$ where $W \in \mathbb{R}^{n\times n}$ is a diagonal matrix of which the diagonal elements are i.i.d. as (25).
Output: Matrix $X' \in \mathbb{R}^{n\times n}$.

The output of Algorithm 6 can be rewritten as $X' = U\Sigma W$. As a corollary of Lemma 7, for any two input matrices X_1 and X_2 which share the same left-singular vectors (same U in their SVD), then as $\tau \to 0$, Algorithm 6 produces an ambiguity set which includes matrices with the same left-singular vectors U.

D Proof of Theorem 2

Proof. Since the selection of each row of W is independent, the high-level structure of the proof is that we first describe such distribution difference in one coordinate of the output and then we compose those differences to derive the final result. It is noted that after SVI, the gap matrix becomes $\hat{X}_1 - \hat{X}_2 = X_1 - X_2 + (\hat{X}_1 - X_1) - (\hat{X}_2 - X_2)$, where the last two quantities need careful control. Now, consider the following identity

$$\hat{X}_2\hat{X}_2^T - \hat{X}_1\hat{X}_1^T = E(X_1+E)^T + X_1E^T. \quad (28)$$

$$\begin{aligned}\boldsymbol{z}^T[(\hat{X}_1\hat{X}_1^T)^{-1} - (\hat{X}_2\hat{X}_2^T)^{-1}]\boldsymbol{z} &= \boldsymbol{z}^T[(\hat{X}_1\hat{X}_1^T)^{-1}(\hat{X}_2\hat{X}_2^T)(\hat{X}_2\hat{X}_2^T)^{-1} - (\hat{X}_2\hat{X}_2^T)^{-1}]\boldsymbol{z}\\ &= \boldsymbol{z}^T[(\hat{X}_1\hat{X}_1^T)^{-1}(\hat{X}_1\hat{X}_1^T + E(X_1+E)^T + X_1E^T)(\hat{X}_2\hat{X}_2^T)^{-1} - (\hat{X}_2\hat{X}_2^T)^{-1}]\boldsymbol{z}\\ &= \boldsymbol{z}^T[(\hat{X}_1\hat{X}_1^T)^{-1}(E(X_1+E)^T + X_1E^T)(\hat{X}_2\hat{X}_2^T)^{-1}]\boldsymbol{z}.\end{aligned} \quad (29)$$

Without loss of generality, replacing $\boldsymbol{z}$ by $X_1\boldsymbol{\alpha}$ in (29), we have

$$\begin{aligned}&\boldsymbol{\alpha}^T[(\hat{X}_1)^{-1}(E(X_1+E)^T + X_1E^T)(\hat{X}_2\hat{X}_2^T)^{-1}]X_1\boldsymbol{\alpha}\\ &= \left[\boldsymbol{\alpha}^T(\hat{X}_1)^{-1}E\right]\cdot\left[X_2^T(\hat{X}_2\hat{X}_2^T)^{-1}X_1\boldsymbol{\alpha}\right] + \left[\boldsymbol{\alpha}^T(\hat{X}_1)^{-1}X_1\right]\cdot\left[E^T(\hat{X}_2\hat{X}_2^T)^{-1}X_1\boldsymbol{\alpha}\right]\\ &= \left[\boldsymbol{\alpha}^T(\hat{X}_1)^{-1}E\right]\cdot\left[X_2^T(\hat{X}_2^{-1})^T\hat{X}_2^{-1}(X_2-E)\boldsymbol{\alpha}\right] + \left[\boldsymbol{\alpha}^T(\hat{X}_1)^{-1}X_1\right]\cdot\left[E^T(X_2^{-1})^T(\hat{X}_2)^{-1}(X_2-E)\boldsymbol{\alpha}\right].\end{aligned} \quad (30)$$

Take a closer look at several key components in (30). The first is $\hat{X}_2^{-1}X_2$. Assume $X_2 = U\Sigma V$ and thus $\hat{X}_2 = U(\Sigma + \tau\cdot\boldsymbol{I})V$. This provides $\hat{X}_2^{-1}X_2 = V^T(\Sigma+\tau\cdot\boldsymbol{I})^{-1}\Sigma V$, where each diagonal element of $(\Sigma+\tau\cdot\boldsymbol{I})^{-1}\Sigma$ is less than 1 since $\tau > 0$. This observation produces a useful corollary that for any vector z,

$\|\hat{X}_2^{-1}X_2 \cdot z\| < \|z\|$. The other useful conclusion we will use is, for arbitrary matrix X, the Frobenius norm is always no less than the l_2 norm. Now, if we rewrite E in the following form that $E = e_i \cdot v_i^T$, where e_i is the i^{th} natural basis, v_i is the i^{th} row of E and i is the index of the sample in which the two adjacent datasets X_1 and X_2 differ. Then, we can further write (30) as

$$\left[\boldsymbol{\alpha}^T(\hat{X}_1)^{-1}e_i\right] \cdot \left[v_i^T \cdot X_2^T(\hat{X}_2^{-1})^T\hat{X}_2^{-1}(X_2 - E)\boldsymbol{\alpha}\right] + \left[\boldsymbol{\alpha}^T(\hat{X}_1)^{-1}X_1 v_i\right] \cdot \left[e_i^T(X_2^{-1})^T(\hat{X}_2)^{-1}(X_2 - E)\boldsymbol{\alpha}\right]. \tag{31}$$

Before proceeding, we first prove the following lemma.

Lemma 8. *Assuming $\boldsymbol{\alpha} \in \mathbb{R}^n$ which is randomly selected from the $(n-1)$-dimensional sphere centered at $\mathbf{0}$ with radius r, i.e., $\|\boldsymbol{\alpha}\| = r$, then $\boldsymbol{\alpha}$ is a Sub-Gaussian random variable and*

$$\Pr(|\boldsymbol{v}^T \cdot \boldsymbol{\alpha}| \leq \frac{r\|v\|\sqrt{-2\log(\delta/4)}}{\sqrt{n + 2\sqrt{n\log(\delta/2)}}}) < 1 - \delta.$$

Proof. Equivalently, we can rewrite $\boldsymbol{\alpha} = t \cdot \boldsymbol{g}/\|\boldsymbol{g}\|$, where $\boldsymbol{g} \sim N(0, \boldsymbol{I}_n)$, since a normalized Gaussian vector is uniformly distributed in the unit $(n-1)$-sphere S_{n-1}. First, for $\|\boldsymbol{g}\|^2$, which follows a χ-distribution, applying the Bernstein inequality, we have the following concentration bound,

$$\Pr(\|\boldsymbol{g}\|^2 \leq n - 2\sqrt{nc}) \leq e^{-c}.$$

On the other hand, $\boldsymbol{v}^T\boldsymbol{g} \sim N(0, \|v\|)$, and thus

$$\Pr(|\boldsymbol{v}^T\boldsymbol{g}| > \|v\|\sqrt{-2\log(\delta'/2)}) < \delta'.$$

Putting the above two bounds together, we have

$$\Pr(r \cdot |\boldsymbol{v}^T\boldsymbol{g}|/\|\boldsymbol{g}\| \leq \frac{r\|v\|\sqrt{-2\log(\delta'/2)}}{\sqrt{n - 2\sqrt{nc}}}) > 1 - \delta' - e^{-c}.$$

Let $c = -\log(\delta/2)$ and $\delta' = \delta/2$, we have the argument claimed.

Back to (31), we have the following observation. First, $\|\hat{X}_1^{-1}e_i\| \leq \frac{1}{\tau}$ since the least singular value of $\hat{X}_1$ is at least τ. Second, $\|v_i^T \cdot X_2^T(\hat{X}_2^{-1})^T\| \leq \|v_i^T\|$.

Applying Lemma 8 on each component of (31), we have that

$$\Pr\left(|\boldsymbol{z}^T[(\hat{X}_1\hat{X}_1^T)^{-1} - (\hat{X}_2\hat{X}_2^T)^{-1}]\boldsymbol{z}| \leq 2(1 + \frac{\|v_i\|}{\tau})\frac{-r^2\|v_i\|2\log(\delta/32)}{\tau(n + 2\sqrt{n\log(\delta/16)})}\right) > 1 - \delta. \tag{32}$$

Now, we can turn to bound the ratio,

$$\frac{P(\hat{X}^{-1}\boldsymbol{z})}{P(\hat{X}_2^{-1}\boldsymbol{z})} = \frac{e^{\psi(\|\hat{X}_1^{-1}\boldsymbol{z}\|^2)}}{e^{\psi(\|\hat{X}_2^{-1}\boldsymbol{z}\|^2)}} \leq e^{L|\|\hat{X}_1^{-1}\boldsymbol{z}\|^2 - \|\hat{X}_2^{-1}\boldsymbol{z}\|^2|} = e^{L|\boldsymbol{z}^T[(\hat{X}_1\hat{X}_1^T)^{-1} - (\hat{X}_2\hat{X}_2^T)^{-1}]\boldsymbol{z}|}. \tag{33}$$

As for the ratio $\frac{det(\hat{X}_1)}{det(\hat{X}_2)}$, we follow the idea in [BBDS12], which is based on the Linskii Lemma [Bha07]:

Lemma 9 ([Bha07]). *Given two matrices X and $X' \in \mathbb{R}^{n\times n}$, whose singular values are $\lambda_1 \geq \lambda_2 \geq ...\lambda_n$ and $\lambda'_1 \geq \lambda'_2 \geq ...\lambda'_n$, respectively, let $E = X' - X$. Then, for every $k \in [1:n]$ and any $1 \leq i_1 < i_2 < ... < i_k \leq n$,*

$$\sum_{j=1}^{k} \lambda_{i_j} \leq \sum_{j=1}^{k} \lambda_{i_j} + \sum_{j=1}^{k} sv(E)_j,$$

where $sv(E)_{[1:n]}$ are the singular values of E in an non-increasing order.

From the above lemma, in our case, E is of rank 1, which has only one non-zero singular value upper bounded by $\mathcal{B}$. Thus, if we can further guarantee the least singular values of both X and X' are at least τ, then

$$\frac{det(X)}{det(X')} = \prod_{i=1}^{n} \frac{\lambda_i}{\lambda'_i} \leq \prod_{i,\lambda_i>\lambda'_i} \frac{\lambda_i}{\lambda'_i} = \prod_{i,\lambda_i>\lambda'_i} (1 + \frac{\lambda_i - \lambda'_i}{\lambda'_i}) \leq e^{\sum_{i,\lambda_i>\lambda'_i} \frac{\lambda_i - \lambda'_i}{\lambda'_i}} \leq e^{\frac{\mathcal{B}}{\tau}}. \tag{34}$$

(32), (33) and (34) together produce the $(\epsilon_0, \delta_0 + \delta_r)$ DP leakage for one dimension of the output, where

$$\epsilon_0 = \frac{\mathcal{B}}{\tau} + 4L(1 + \frac{\mathcal{B}}{\tau}) \frac{-r^2 \mathcal{B} \log(\delta_0/32)}{\tau(n + 2\sqrt{n \log(\delta_0/16)})}.$$

Applying the well-known DP composition [DRV10], a d'-fold of $(\epsilon_0, \delta_0 + \delta_r)$-DP mechanism will produce $\big(d'\epsilon_0(e^{\epsilon_0} - 1) + \epsilon_0\sqrt{2d' \log(1/\tilde{\delta})}, d'(\delta_0 + \delta_r) + \tilde{\delta}\big)$, and the claim follows.

E Proof of Theorem 3

Proof. As mentioned before, $I(W^{-1}, \sigma^{-1}; Adv) \geq I(W^{-1}; Adv|\sigma^{-1})$, where conditional on σ^{-1}, Adv is equivalent to satisfying

$$\Pr_{\bar{s}\sim s_a+s_p,W}(\|Adv(\bar{s}W) - \bar{s}\| < \epsilon) \geq 1 - \delta.$$

Thus, we first construct a packing set over the support set of W^{-1}, i.e., $[-\beta, \beta]^{d\times d}$. In the following, we use $\|V\|$ to define the norm of a matrix V as the largest eigenvalue of V.

Lemma 10. *For a β-uniform matrix V, with probability at least $(1 - 2d \cdot exp(-\frac{c^2}{2d^2\beta^2}))$, $\|V\| < c$.*

Proof. It is well-known that a uniform distributed variable is sub-Gaussian. To be specific, if $x \sim U[-\beta, \beta]$, then $\mathbb{E}_x[e^{\lambda x}] \leq e^{\lambda^2\beta^2/2}$. Therefore, consider a d-dimensional uniform vector $\boldsymbol{v}$ whose entries are i.i.d. selected from $[-\beta, \beta]$, and any d-dimensional unit vector $\boldsymbol{u}$ such that $\|\boldsymbol{u}\| = 1$. Applying the Hoeffding bound on sub-Gaussian variables, we have

$$\Pr(|\langle v, u\rangle| \geq t) \leq 2exp(-\frac{t^2}{2\beta^2}).$$

Setting $t = \frac{c}{\sqrt{d}}$, with a union bound on the above, we have, if V is selected to be a $d \times d$ β-uniform matrix,

$$\Pr(\|uV\| < c) \geq 1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2}).$$

Therefore, with probability at least $(1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2}))$, $\|V\| < c$.

With Lemma 10, a corollary is that, with probability at least $(1-2d\cdot exp(-\frac{c^2}{2d^2\beta^2}))$, the smallest eigenvalue of V^{-1} should be at least $1/c$. Let $W = V^{-1}$. Now we can move to lower bound $\|W(W_1^{-1} - W_2^{-1})\|$. In the next step, we consider the anti-concentration of a Gaussian distribution. Assuming two vectors $u_1, u_2 \in \mathbb{R}^d$, such that $\|u_1 - u_2\| \geq \Delta$, then with probability at least $(1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2}))$, $\|W(u_1 - u_2)^T\| \geq \Delta/c$. For any $t > 0$ and a Gaussian variable x distributed as $N(0, \tau)$, $\Pr(|x| > t) = 2(1 - \Phi(t/\sqrt{\tau}))$, where $\Phi(t)$ is the cumulative distribution function of the normal distribution $N(0, 1)$, i.e., $\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-t^2/2} dt$. Therefore, by setting $\tau = \tau_a + \tau_b$, for a Gaussian vector $\bar{s} \in \mathbb{R}^d$ of which each entry is i.i.d. selected from $N(0, \tau)$, and any $t > 0$, we have

$$\Pr(|\langle \bar{s}, W(u_1 - u_2)^T \rangle| \geq t) \geq 2(1 - \Phi(tc/\sqrt{\tau}\Delta))(1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2}). \tag{35}$$

After the above preparation, we can now show how to construct a packing set w.r.t. W^{-1}. One candidate is simply a partition on the interval $[-\beta, \beta]$. We naturally divide $[-\beta, \beta]$ into $2\beta/\Delta$ intervals of equal length Δ. For simplicity, we assume β/Δ is an integer. Then, we select one half of them in a form: $\{[-\beta + 2i \cdot \Delta, -\beta + (2i + 1) \cdot \Delta], i = 0, 1, ..., \beta/\Delta - 1\}$. It is noted that arbitrary different elements from different sets are of distance at least Δ. Now, we generalize such a partition to each entry of W^{-1} within the range $[-\beta, \beta]^{d\times d}$. The concatenated partition is in a form $\{[-\beta + 2i \cdot \Delta, -\beta + (2i + 1) \cdot \Delta], i = 0, 1, ..., \beta/\Delta - 1\}^{d\times d}$, which in total produces $(\beta/\Delta)^{d^2}$ many subclasses. For arbitrary W_1^{-1} and W_2^{-1} from two different subclasses, we know that the l_2 norm of at least one column of $(W_1^{-1} - W_2^{-1})$ is at least Δ. Thus, we finally have that for any W_1^{-1} and W_2^{-1} from two different subclasses described above, with probability at least $\delta = 2(1 - \Phi(t\sqrt{c}/\sqrt{\tau}\Delta))(1 - 2d \cdot exp(-\frac{c^2}{2d\beta^2})$, $\|\bar{s}W(W_1^{-1} - W_2^{-1})\| \geq t$. Hence, if we select one element from each subclass, they naturally produce a (t, δ) packing set w.r.t. W^{-1} provided the transformed data $\bar{s}W$.

A simple but useful fact which will be used later is, such a packing set construction can also lead to a straightforward covering of the support set $[-\beta, \beta]^{d\times d}$. In other words, we divide the full space $[-\beta, \beta]^{d\times d}$ into a union of packing sets in a form

$$(\{[-\beta+2i\cdot\Delta, -\beta+(2i+1)\cdot\Delta], i = 0, 1, ..., \beta/\Delta-1\}, \{[-\beta+(2i+1)\cdot\Delta, -\beta+(2i+2)\cdot\Delta], i = 0, 1, ..., \beta/\Delta-1\})^{d\times d}.$$

The following analysis will connect the mutual information and its partition form.

Consider

$$I(x, y) = \sum_{x\in\mathcal{B}, y\in\mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)},$$

where there exist two disjoint Borel sets $\mathcal{B}_1$ and $\mathcal{B}_2$ as a partition of $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$. We have the following identity,

$$\sum_{x\in\mathcal{B}_1, y\in\mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} = -\Pr(x \in \mathcal{B}_1) \sum_{x\in\mathcal{B}_1, y\in\mathcal{Y}} \frac{p(x, y)}{\Pr(x \in \mathcal{B}_1)} \log \frac{p(x)p(y)}{p(x, y)}. \tag{36}$$

On the other hand, it is noted that when we restrict x into $\mathcal{B}_1$, the conditional mutual information $I(x; y|x \in \mathcal{B}_1)$ equals

$$\sum_{x\in\mathcal{B}_1, y\in\mathcal{Y}} p(x, y|x \in \mathcal{B}_1) \log \frac{p(x, y|x \in \mathcal{B}_1)}{p(x|x \in \mathcal{B}_1)p(y)} = \sum_{x\in\mathcal{B}_1, y\in\mathcal{Y}} \frac{p(x, y)}{\Pr(x \in \mathcal{B}_1)} \log \frac{p(x, y) \Pr(x \in \mathcal{B}_1)}{p(x) \Pr(x \in \mathcal{B}_1)p(y)}, \tag{37}$$

which is equivalent to the right hand side of (36) multiplying a factor $1/\Pr(x \in \mathcal{B}_1)$. Through the above discussion, we can focus on the conditional mutual information on each packing set designed.

It is well known that under Shannon entropy, Fano's inequality can produce an upper bound on $H(T^{-1}|Adv)$. However, Fano's inequality cannot directly apply to the continuous case. To overcome this, we propose a generalization under proper assumptions.

Lemma 11 (A Generalization of Fano's Inequality). *Let χ be uniformly distributed over a domain (possibly continuous) $\mathcal{X}$ where there exists a partition $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup ... \cup \mathcal{X}_N$. $\{\mathcal{X}_i, i = 1, 2, ..., N\}$ are disjoint and further $\Pr(\chi \in \mathcal{X}_i) = \frac{1}{N}$. Then, for any estimator $\hat{\chi}$ which satisfies a Markov Chain $\chi \to y \to \hat{\chi}$ if there exists a sequence of discretizations $D_1, D_2, ...$ of $\mathcal{X}$ such that $\lim_{i\to\infty} D_i = \mathcal{X}$, then*

$$I(\chi; \hat{\chi}) \geq (\log N)(1 - P_e) - 1,$$

where P_e denotes the probability that χ and $\hat{\chi}$ do not belong to the same set $\mathcal{X}_i$.

Proof. Let E be an indicator that $E = 1$ if both $\hat{\chi}$ and χ belong to the same set $\mathcal{X}_i$; otherwise $E = 0$. Consider a discretization where each partition set $\mathcal{X}_i$ contains C elements and χ is uniformly distributed over those discrete elements. Since $I(\chi; \hat{\chi}) = H(\chi) - H(\chi|\hat{\chi}) = \log(NC) - H(\chi|\hat{\chi})$, in the following, we focus on $H(\chi|\hat{\chi})$. Applying the Chain rule,

$$H(E, \chi|\hat{\chi}) = H(\chi|\hat{\chi}) + H(E|\chi, \hat{\chi}) = H(E|\hat{\chi}) + H(\chi|E, \hat{\chi}). \tag{38}$$

It is noted that E is deterministic given χ and $\hat{\chi}$, and thus $H(E|\chi, \hat{\chi}) = 0$. On the other hand, $H(E|\hat{\chi}) \leq H(E) \leq 1$ and

$$H(\chi|E, \hat{\chi}) = \Pr(E = 0)H(\chi|\hat{\chi}, E = 0) + \Pr(E = 1)H(\chi|\hat{\chi}, E = 1) \leq (1 - P_e) \log C + P_e \log NC.$$

Thus, combining all, we know

$$H(\chi) - H(\chi|\hat{\chi}) \geq \log(NC) - 1 - \log C - P_e \log(NC/C) = (1 - P_e) \log N - 1. \tag{39}$$

Now, we will apply Lemma 11 by replacing $\chi, \hat{\chi}$ and N by W^{-1}, Adv and the number of packing sets in (39). Please note that Adv may not necessarily simply be an estimation of W^{-1}, i.e., we do not restrict Adv within the domain of $[-\beta, \beta]^{d\times d}$. But virtually, for any Adv operator, there exists at most one (ϵ, δ)-statistically close W^{-1} in a $(2\epsilon, 2\delta)$ packing set. Let $P_e = 0$, and thus we obtain a lower bound on the conditional mutual information $I(W^{-1}; Adv|\sigma^{-1}, W^{-1} \in C_{2\epsilon,2\delta})$, where $C_{2\epsilon,2\delta}$ denotes one $(2\epsilon, 2\delta)$ packing set constructed above. As assumed, W^{-1} is uniformly distributed either across the full support set $[-\beta, \beta]^{d\times d}$, and thus also uniformly across one packing set. Recall the connection between mutual information and the equivalent form of the sum of conditional mutual information developed in (36) and (37), we know such a bound is also an lower bound for $I(W^{-1}; Adv|\sigma^{-1})$. The theorem follows.

F Proof of Theorem 4

Proof. To derive an upper bound, it is noted that $I(W^{-1}, \sigma^{-1}; s_a, T(s_a + s_p)) = H(s_a, T(s_a + s_p)) - H(s_a, T(s_a + s_p)|W^{-1}, \sigma^{-1})$. Clearly, $H(s_a, T(s_a + s_p)) \leq H(s_a) + H(T(s_a + s_p))$. On the other hand, $H(s_a, T(s_a+s_p)|W^{-1}, \sigma^{-1}) = H(T(s_a+s_p)|W^{-1}, \sigma^{-1})+H(s_a|T(s_a+s_p), W^{-1}, \sigma^{-1})$. As assumed, the public data s_a is a Gaussian vector, where each entry i.i.d. follows $N(0, \tau_a)$. Thus, $H(s_a) = \frac{d}{2}(1+log(2\pi\tau_a))$. As for $H(s_a|T(s_a + s_p), W^{-1}, \sigma^{-1})$, note that, once $T(s_a + s_p)$ and (W^{-1}, σ^{-1}) are given, they also

determine $s_a + s_p$ uniquely. For simplicity, we use T^{-1} to denote the (W^{-1}, σ^{-1}) in the following. Therefore, we have

$$\begin{aligned} H(s_a|T(s_a+s_p), T^{-1}) &= \int_{s' \in \mathbb{R}^d, t' \in [-\beta,\beta]^{d^2}} p(T(s_a+s_p)=s', T^{-1}=t') H(s_a|T(s_a+s_p)=s', T^{-1}=t') \\ &= \int_{s \in \mathbb{R}^d} p(s_a+s_p=s) H(s_a|s_a+s_p=s) = H(s_a|s_a+s_p). \end{aligned} \tag{40}$$

Substituting the distribution of s_a and s_p into the above equation, we have

$$\begin{aligned} H(s_a|s_a+s_p) &= -\int_{s_a}\int_{s_a+s_p} p(s_a, s_a+s_p)\log(p(s_a|s_a+s_p))ds_a\, ds_p \\ &= -d\int_x\int_y \phi_{\tau_a}(x)\phi_{\tau_p}(y-x)\log\frac{\phi_{\tau_a}(x)\phi_{\tau_p}(y-x)}{\phi_{\tau_a+\tau_p}(y)}dx\, dy \\ &= -d\int_x\int_y \phi_{\tau_a}(x)\phi_{\tau_p}(y-x)\big(\log\phi_{\tau_a}(x)\phi_{\tau_p}(y-x) - \log\phi_{\tau_a+\tau_p}(y)\big)dx\, dy. \end{aligned} \tag{41}$$

Here, ϕ_τ denotes the density function of the Gaussian distribution $N(0,\tau)$, i.e., $\phi_\tau(x) = \frac{1}{\sqrt{2\pi\tau}}e^{-x^2/2\tau}$. We separately calculate the two quantities in (41).

As a straightforward corollary of the differential entropy of a Gaussian, since $\int_y \phi_{\tau_b}(y-x)dy = 1$ for fixed x, we have

$$-\int_x\int_y \phi_{\tau_a}(x)\phi_{\tau_p}(y-x)\log\phi_{\tau_a}(x)\phi_{\tau_p}(y-x)dx\, dy = \frac{1}{2}(1+\log 2\pi\tau_p) + \frac{1}{2}(1+\log 2\pi\tau_a). \tag{42}$$

For the very last part, $-\int_x\int_y \phi_{\tau_a}(x)\phi_{\tau_p}(y-x)\log\phi_{\tau_a+\tau_p}(y)$, which is equivalent to

$$-\int_y\big(\int_x \phi_{\tau_a}(x)\phi_{\tau_p}(y-x)dx\big)\cdot\log\phi_{\tau_a+\tau_p}(y)dy = \frac{1}{2}(1+\log 2\pi(\tau_a+\tau_p)). \tag{43}$$

Therefore, the sum of (42) and (43) produces a closed form for $H(s_a|T(s_a+s_p), T^{-1})$. Since $H(s_a) = \frac{d}{2}(1+\log 2\pi(\tau_a))$, we have $H(s_a) - H(s_a|T(s_a+s_p), T^{-1}) = \frac{d}{2}(\log(\frac{\tau_a+\tau_p}{\tau_p}))$, which approaches 0 as $\tau_p \to \infty$. Finally, for $H(T(s_a+s_p)) - H(T(s_a+s_p))|T^{-1})$, since $T(s_a+s_p)$ is restricted to a finite discrete set $\mathcal{K}$, it is upper bounded by $\log(|\mathcal{K}|)$. Combining both, the theorem follows.

G Proof of Theorem 5

Proof. With a similar reasoning as the proofs to Theorem 3 and Theorem 4, the following content includes two parts, where the first derives a lower bound $I(W; Adv)$, while the second derives an upper bound of $I(W; s_a, T(\bar{s}))$.

We first construct a packing set of W. For each row $w \in \mathbb{R}^d$, which is within an l_2 ball of radius β denoted by $\mathcal{B}_\beta$ we split the support set into $2\frac{\beta}{\Delta}$ parts based on the norm in a form $S_1, S_2, ..., S_{2\frac{\beta}{\Delta}}$, where $S_i = \{w | \|w\| \in ((i-1)\cdot\frac{\Delta}{2}, i\cdot\frac{\Delta}{2}]\}$. Similarly, a partition over $(\mathcal{B}_\beta)^d$ can be $\{S_{[1:2\frac{\beta}{\Delta}]}\}^d$. The following lemma develops the foundation of the packing set.

Lemma 12. *For $\bar{s} \in N(0, \tau \cdot \boldsymbol{I}_d)$ and any $W_1, W_2 \in \mathbb{R}^{d \times d'}$ such that the norm of each column of $W_1 - W_2$ is at least $\frac{\Delta}{2}$, then*

$$\Pr(\|\bar{s}(W_1 - W_2)\| \geq 2\epsilon) \geq 2\delta,$$

where ϵ and δ satisfy the following

$$\delta = \frac{1}{2}(1 - d'(2\Phi(\frac{2\sqrt{2}\epsilon}{\sqrt{d'\tau}\Delta}) - 1)).$$

Proof. Since the inner product $\langle \bar{s}, w \rangle \sim N(0, \tau\|w\|)$. Thus, $\Pr(|\langle \bar{s}, w \rangle| \geq t) = 2(1 - \Phi(t/\sqrt{\tau\|w\|}))$. By setting $t = \frac{2\epsilon}{\sqrt{d'}}$ and $\|w\| = \frac{\Delta}{2}$, with a union bound we have

$$\Pr(\|\bar{s}(W_1 - W_2)\| \geq 2\epsilon)\|) \geq 1 - d'(2\Phi(\frac{2\sqrt{2}\epsilon}{\sqrt{d'\tau}\Delta}) - 1).$$

Hence, by only selecting non-adjacent partition subsets, $(\{S_{2i+1}, i = 0, 1, ..., \frac{\beta}{\Delta} - 1\}, \{S_{2i}, i = 1, 2, ..., \frac{\beta}{\Delta}\})^{d'}$ produces a covering of the support domain of W while on the other hand, if we select one element from each subclass, they will form a packing set. Following Lemma 11, with the same notations used in Lemma 12, we have

$$I(W; Adv) \geq d' \log(\beta/\Delta) - 1.$$

In the following, we focus on upper bounding $I(W; s_a, T(\bar{s}))$, where we have

$$\begin{aligned} I(W; s_a, T(\bar{s})) &= H(s_a, T(\bar{s})) - H(s_a, T(\bar{s})|W) \\ &\leq H(s_a) + H(T(\bar{s})) - H(s_a|W) - H(T(\bar{s})|s_a, W). \end{aligned} \tag{44}$$

Here, it is noted that the selection of s_a is independent of that of W, and thus, $H(s_a|W) = H(s_a)$. On the other hand, provided W and s_a, $T(\bar{s}) = T(s_a + s_p)$ also uniquely determines $s_p W$. Therefore, (44) can be further rewritten as

$$I(W; s_a, T(\bar{s})) \leq H(T(\bar{s})) - H(s_p W|W). \tag{45}$$

To analyze $T(\bar{s})$, for w such that $\|w\| \sim U(1, 1 + \beta]$, let $x = \langle \bar{s}, w \rangle$, we have

$$p(x = a) = \int_1^{1+\beta} \frac{1}{\beta} \cdot \frac{e^{-a^2/(2\tau\|\omega\|^2)}}{\sqrt{2\pi\tau}\|\omega\|} d\|\omega\| = \frac{1}{2\sqrt{2\pi\tau}\beta}\big(\Gamma(0, \frac{a^2}{2\tau(\beta+1)^2}) - \Gamma(0, \frac{a^2}{2\tau})\big).$$

Here, $\Gamma(x, y)$ is the upper incomplete gamma function where $\Gamma(x, y) = \int_y^\infty t^{x-1}e^{-t}dt$. Thus, we have

$$H(T(\bar{s})) = -2\int_0^\infty \frac{1}{2\sqrt{2\pi\tau}\beta}\big(\Gamma(0, \frac{a^2}{2\tau(\beta+1)^2}) - \Gamma(0, \frac{a^2}{2\tau})\big) \log\Big(\frac{1}{2\sqrt{2\pi\tau}\beta}\big(\Gamma(0, \frac{a^2}{2\tau(\beta+1)^2}) - \Gamma(0, \frac{a^2}{2\tau})\big)\Big)\, da$$

As for $H(\langle s_p, w \rangle|w)$, we have

$$\begin{aligned} H(\langle s_p, w \rangle|w) &= \int_\omega p(\omega) H(\langle s_p, w \rangle)|\|w\| = \omega) = \frac{1}{\beta}\int_0^\beta H(\tilde{s} \sim N(0, \tau \cdot \omega^2)) d\omega \\ &= \frac{1}{2\beta}\int_1^{1+\beta} (\log(2\pi e\tau_p) + 2\log\omega)\, d\omega = \frac{\log(2\pi e\tau_p)}{2} + \frac{\beta+1}{\beta}\log(\beta+1) - 1. \end{aligned} \tag{46}$$

With (46), let the function $\Psi(\beta, \tau)$ be in a form,

$$\Psi(\beta,\tau) = -\int_0^\infty \frac{1}{\sqrt{2\pi\tau\beta}}\big(\Gamma(0, \frac{a^2}{2\tau(\beta+1)^2}) - \Gamma(0, \frac{a^2}{2\tau})\big) \log\big(\frac{1}{2\sqrt{2\pi\tau\beta}}\big(\Gamma(0, \frac{a^2}{2\tau(\beta+1)^2}) - \Gamma(0, \frac{a^2}{2\tau})\big)\big)\, da,$$

and we can upper bound $I(W; s_a, T(\bar{s}))$ as follows.

$$I(W; s_a, T(\bar{s})) \leq d'[\Psi(\beta, \tau_a + \tau_p)) - (\frac{\log(2\pi e\tau_p)}{2} + \frac{\beta+1}{\beta}\log(\beta+1) - 1)]. \tag{47}$$

Combining both the lower bound of numerator and the upper bound of the denominator, we have the claim.