The Art of Labeling: Task Augmentation for Private (Collaborative) Learning on Transformed Data

Hanshen Xiao
MIT
hsxiao@mit.edu

Srinivas Devadas
MIT
devadas@mit.edu

ABSTRACT

We tackle the problems of private learning where an owner wishes to outsource a training task to an honest-but-curious server while keeping its data private, and private collaborative learning where two (or more) mutually distrusting owners outsource respective training data sets to an honest-but-curious server while keeping their data sets private from the server and each other.

The privacy property we provide is information-theoretic in nature, Probably Approximately Correct (PAC) approximation resistance (abbreviated to PAC security). Each owner transforms its data and labels using a private transform. The server combines samples from each data set into expanded samples with corresponding expanded labels – we refer to this step as *Task Augmentation*. The server can be used for inference by any owner by sending it transformed samples. Unlike most prior approaches, our transformed data approach maintains privacy for each entity, even in the case where the server colludes with *all* other entities. Importantly, we show the utility of collaborative learning typically exceeds the utility that can be achieved by any entity restricted to its own data set.

Another important application we show is that the Task Augmentation approach can also be used in the single owner case by adding *labeled, learnable noise* to amplify privacy. This can be straightforwardly used to produce (Local) Differential Privacy ((L)DP) guarantees. We show that adding labeled noise as opposed to a conventional (L)DP additive noise mechanism significantly improves the privacy-utility tradeoff in private learning under the same setup.

CCS CONCEPTS

• **Security and privacy** → Information-theoretic techniques;

KEYWORDS

privacy; machine learning; collaborative learning

1 INTRODUCTION

With the great success of machine learning, the privacy of data processing is receiving increasing attention, both in the two-party setting where a data owner wishes to outsource training and/or inference to an untrusted, typically honest-but-curious server, and in a multi-party (or collaborative) setting where multiple data owners wish to outsource learning tasks while exploiting aggregation of individual data sets.

There are many classes of approaches, each with strengths and limitations, and which provide differing privacy guarantees. Approaches based on partial or Fully Homomorphic Encryption [20, 41, 42], Garbled Circuits [22] and combinations provide strong computational security guarantees but also suffer large computational overheads, restricting their use to small-scale problems in the two-party setting. Approaches to collaborative multi-party learning such as Federated Learning are efficient but require placing trust in an aggregating server [48]. Differential Privacy [7, 15, 16, 31] based approaches can be efficient both in the two-party and multi-party settings, but unlike the two previous classes of approaches, they come with a privacy-utility tradeoff in that greater privacy (smaller ϵ) typically results in lower utility. This is especially true in Local DP (LDP) [11] settings: when owners decide to release the data (rather than responding to queries) and resort to outside computing services, LDP becomes the only known and widely-applied privacy metric. Unfortunately, even to preserve a single attribute, LDP comes with a constant noise on each data entry, which most medium size learning tasks cannot afford. Thus, the tradeoff between reasonable security and utility remains a long-standing challenge.

Recently, approaches based on data transformation have been proposed, e.g., Instahide [25], Dauntless [47]. In these approaches, the owner transforms its data using a private transform, and the transformed data is sent to the untrusted (honest-but-curious) server. The challenge in this approach is twofold: First, a privacy property for the exposed transformed data has to be shown, and second, the utility of learning on the transformed data is expected to match the utility of non-private learning. In this paper, we utilize the Dauntless framework [47] of Probably Approximately Correct (PAC) approximation (or inference) resistance, abbreviated to PAC security, and contribute simpler and more powerful data transformation techniques for both single and multiple owner settings, as well as associated proofs that relate PAC security (measured by lower bounds on the number of exposed transformed samples) to the entropy of private samples.

We present a novel *Task Augmentation* approach where we train a model for a more general task, in which the original learning task(s) become subproblem(s). Given a c-classification problem on a data set (x_i, y_i)[1], and another data set corresponding to a possibly unrelated c'-classification task, (x'_i, y'_i), we define a new $c \times c'$-classification problem by considering the Cartesian product of any pairs $\{(x_i, y_i), (x_j, y_j)\}$ in a form $\big((x_i|x'_j), (y_i|y'_j)\big)$. The composite feature domain is now within the $(d + d')$-dimensional space while the label domain also increases correspondingly to $c \times c'$ classes. In the case of private collaborative learning, the first owner provides a *privately transformed* data set for a c-classification task, and a second owner provides a different transformed data set, using a different private transform, for a c'-classification task. Task Augmentation is performed by the server, which cannot "see" the original data or labels, followed by training. Once the server has the trained model, an owner can outsource inference on a new sample by providing the server with an appropriately transformed sample. Alternately, the trained model can be returned to any or all owners, who can

[1]Here, feature $x_i \in \mathbb{R}^d$ while label $y_i \in \mathbb{R}^c$ is a one-hot vector.

run inference locally by transforming their samples and using one (or more) transformed samples from each of the other owners. We note that there is no exposure in either training or inference of private (non-transformed) samples. We show that Task Augmentation provides utility benefits in collaborative learning, i.e., the utility of models trained on task augmented data is higher than the utility of models obtained from individual data sets, and approaches the utility of non-private collaborative training. Our approach does *not* require non-collusion assumptions across owners and the server as in many multi-party computation schemes. We further note that the Task Augmentation approach may be of independent interest outside of privacy considerations to improve model generalization in machine learning, though this is not the focus of this paper.

In summary, this paper makes the following contributions:

(1) We provide a new PAC security theoretical result based on the PAC security definition of [47], which corresponds to a simpler transform (multiplication by a random matrix as opposed to the potentially unstable inverse of a random matrix), and which has a precise and parameterizable quantification of a sample lower bound.
(2) While the transform of Dauntless [47] only worked well for fully-connected networks, we present private transforms and associated PAC security proofs that work well for Transformer networks and Convolutional Neural Networks.
(3) We show how to achieve private collaborative learning using a Task Augmentation approach. Crucially, we do not require a non-collusion assumption between mutually distrusting users and a honest-but-curious server.
(4) We propose a novel way to introduce *labeled* noise incorporated with Task Augmentation for further privacy amplification. Such a technique can be straightforwardly applied to produce (L)DP guarantees but with a sharpened utility-privacy tradeoff.
(5) We provide experimental results on the MNIST and CIFAR-10 data sets for private learning and private collaborative learning that show that a private transformation approach can achieve utility, efficiency and provable security.

The rest of the paper is organized as follows. We review the Dauntless framework [47] in Section 2. New private transforms and proofs are the subject of Section 3, where we present a simple, generic transform for fully connected networks, and show how to augment it for Transformer networks and Convolutional Neural Networks (CNNs). We describe the Task Augmentation approach in Section 4. Then, we present a natural private collaboration scheme based on Task Augmentation and private data transformation in Section 5. In Section 6, we describe how labeled noise is used to generate an auxiliary learning task to augment privacy. In Section 7, we provide two sets of results on private collaborative learning and private learning for image recognition problems, including MNIST and CIFAR-10. Related work is the subject of Section 8. We conclude in Section 9.

Notations: We use $\mathcal{I}(a, b)$ to denote the mutual information between two random variables a and b. $\mathcal{H}(a)$ is the entropy of a variable a and the following relationship holds $\mathcal{I}(a, b) = \mathcal{H}(a) - \mathcal{H}(a|b)$. We use $\|W\|$ to denote the l_2 norm of a matrix W, i.e., $\|W\| = \sup_{\|x\|=1} \|Wx\|$ and $\|W\|_F$ to denote the Frobenius norm of W.

2 SUMMARY OF DAUNTLESS FRAMEWORK

We begin with the main insights underlying existing security metrics. In general, either cryptographic encryption, such as RSA, or an information theory based Differentially Private additive noise mechanism, can be regarded as a transformation T on plaintext x. To measure the security or privacy guarantee with respect to (w.r.t.) the transformation, dating back to Shannon's perfect secrecy [40], a primary approach is to measure the *additional information* provided by the ciphertext $T(x)$ (transformed data). Said another way, for any adversarial prior knowledge assumed regarding the plaintext x, after the observation of the ciphertext $T(x)$, the difference between the prior and posterior opinions should be limited. If, for a polynomial-time computationally-bounded adversary, such difference is negligible, then we say the mechanism T is cryptographically secure. From an information-theoretical standpoint, for example, Differential Privacy (DP), such difference (restricted to an individual data point) is captured by parameters $e^{\epsilon}, (\delta)$ in $(\epsilon, (\delta))$-DP [16]. When $\epsilon = 0$, DP is equivalent to perfect secrecy, where the plaintext and ciphertext are independent.

In the Dauntless framework [47], a different security definition is proposed, termed PAC security, that takes into account the prior knowledge of the adversary (see Section 2.1). We describe characteristics of a good transformation in Section 2.2, and consider the use of neural networks to transform data in Section 2.3. We review the Dauntless proof strategy for PAC security in Section 2.4, and contrast it with encryption approaches in Section 2.5.

2.1 PAC Security

PAC learning theory is mimicked 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 2.1 (Resistance to (ϵ, δ)-PAC Approximation [47]). A transformation mechanism $\mathcal{M}(X, \theta)$ on a data set X of m data points, where $\theta \in \Theta$ is the random seed, is resistant to (ϵ, δ)-PAC approximation for private input data X in a distribution $\mathcal{P}^n$, if given the output $\mathcal{M}(X)$, there does not exist an (possibly computationally-unbounded) algorithm which returns an estimator of the inverse of the mechanism $\mathcal{M}$ based on $\mathcal{M}(X, \theta)$, namely, $g(\cdot) \in \{\mathcal{M}^{-1}(\cdot, \theta), \theta \in \Theta\}$, [2] such that

$$\Pr_{x\sim\mathcal{P}, X\sim\mathcal{P}^n, \theta}\left(\|g(\mathcal{M}(x, \theta)) - x\| < \epsilon | \mathcal{M}(X, \theta)\right) \geq 1 - \delta. \quad (1)$$

The data distribution $\mathcal{P}$ captures the prior knowledge from the adversary's view. If we take the mechanism $\mathcal{M}$ as an encryption scheme with the key θ randomly generated, (ϵ, δ)-PAC approximation resistance captures the hardness in determining or approximating the key (or true inverse function) given exposed data. In the above definition, we assume that in the adversarial prior knowledge, data points are distributed independently. The above metric can also be generalized to measure the approximation error on data points distributed differently. It is also worth mentioning that the adversarial recovery is determined both by the additional information provided

[2] When $\mathcal{M}(\cdot, \theta)$ is not invertible, we may approximate each $\mathcal{M}^{-1}(\cdot, \theta)$ by some function $\hat{\mathcal{M}}^{-1}(\cdot, \theta)$ that $\Pr_{x\sim P}(\|\hat{\mathcal{M}}^{-1}(\mathcal{M}(x, \theta), \theta)) - x\| < \epsilon) \geq 1 - \delta$.

by ciphertext and the adversary's prior knowledge on plaintext.[3] PAC security suggests an impossibility result even for an unbounded adversary to recover the secret input within ϵ-error under *restricted* prior knowledge.

An intuitive interpretation for the role of prior knowledge in the security analysis is analogous to Schrödinger's cat. Imagine that one wants to randomly publish an image from a secret data set, containing one million image samples. Before publishing, the state of whether a particular image will be published is stochastic (a hypothetical cat in a box), but once published (box is opened and observation occurs), uncertainty collapses to a definite result. However, different from exposing the image directly, with a good design of the transformation, sufficient amount of uncertainty will be preserved (rather than collapsed) even after the observation occurs on the transformed image. Thus, if we view the setup of DP or Local DP (LDP) from a prior knowledge standpoint, though the focus is on the participation of an individual data point, the setup is indeed strong, where the adversary's prior knowledge is sufficient to determine almost the full data set except for one data point. As will be shown below, the PAC security model provides a more generic framework to handle different prior knowledge setups.

In the context of private learning, suppose a data owner holds and decides to expose n samples denoted by $\mathcal{S} = \{s_i = (x_i, y_i), i = 1, 2, ..., n\}$. Here, we consider a uniform transformation T across the full data set $\mathcal{S}$, where the transformed samples become $T(\mathcal{S}) = \{T(s_i), i = 1, 2, ..., n\}$. In general, the correspondence between s_i and $T(s_i)$ is not provided to the adversary, as most learning procedures do not require a specific order of the samples. To further strengthen the adversary, we assume such correspondence is known and in such a setup, resistance to adversary inference using PAC security can be described as follows. For simplicity, we assume the prior knowledge w.r.t. each sample is identical as each s_i is i.i.d. selected from some distribution P and the transformation T is randomly generated from a distribution Q. Then, PAC security is equivalent to determining the maximal number m of transformed samples that can be exposed given P, Q and security parameters ϵ, δ, i.e., there does not exist an inversion estimator Adv such that

$$\Pr_{x\sim P, T\sim Q, \mathcal{S}\sim P^m}(\|Adv(T(x), T(\mathcal{S})) - x\| < \epsilon) \geq 1 - \delta. \quad (2)$$

Here $Adv(T(x), T(\mathcal{S}))$ denotes that Adv takes $T(x)$ as input to recover x while the algorithm is developed on observation $T(\mathcal{S})$. In [47], it is pointed out that one can augment the entropy of $\mathcal{S}$ by mixing private and public data. The above description can also be generalized to the case where an adversary has different prior knowledge on each s_i, for example, $s_i \sim \mathcal{N}(\mu_i, \tau \cdot \boldsymbol{I})$, a multivariate Gaussian of mean μ_i.

2.2 Transformation Design

Ideally, a good transformation is expected to provide good privacy, regarding the original data, and good utility, regarding the model trained on the transformed data. Arguably, the most natural transformation would be perturbation, such as the common Gaussian or Laplace Mechanism in DP. Nonetheless, when the model over transformed samples is also for private use, there is much more freedom in the choice of transformation. In our private (collaborative) learning scenario, once a model $f(\cdot)$ has been trained on transformed samples, using a transformation T, inference on the model $f(\cdot)$ for a new sample x_{new} *also requires transformation*, i.e., we evaluate $f(T(x_{new}))$.

In general, any continuous function with good locality can be a candidate, where the original sample domain can be smoothly transformed. However, as we will show later, the resultant performance of a transformation even for the same data set varies significantly with training mechanisms. This raises a challenging and practical question that if experimentally transformed data is not efficiently learnable by a training mechanism, is such a failure due to the transformed data being computationally hard to learn or is it due to an improper selection of training mechanism? In the following, we first present tenets of transformation design.

In a statistical learning viewpoint, assume that f^* is the optimal model where $f^* = \arg\min_f \mathbb{E}_{(x,y)} l(f(x), y)$, where $l(\cdot, \cdot)$ is some loss function and (x, y) denotes the sample from some distribution. Imagine transformed samples $(T(x), y)$ for some transformation T, to which the optimal model becomes $f \circ T^{-1}$. Here, we simply assume T^{-1}, or that T^{-1} is an approximation of the inverse of T. To provide reasonable utility guarantees, a necessary condition is that $f \circ T^{-1}$ is approximatable by the training algorithm applied. To match that, one can always creatively modify the existing training model or even propose something new, which will be suitable for particular transformed data. A conservative strategy employed in the Dauntless framework [47] is to force the transformation to match existing training mechanisms, which may directly benefit from machine learning research advances, and the training mechanisms can be taken as a black box. This framework can be outlined as follows: for given data $\mathcal{D}$ from some distribution $\mathcal{P}$, if there exists a training mechanism which can find a model from a function set C, then we select a transformation T such that for any function $c \in C$, $c \circ T^{-1} \in C$.

2.3 Neural Networks

The well-known uniform approximation theorem states that any continuous function over a compact set can be approximated arbitrarily closely by a neural network with sufficiently large width and a good choice of weights. In general, an L-layer feed-forward neural network $\mathcal{N}(x)$ can be expressed as

$$\mathcal{N}(x) = \sigma^{(L)}(\sigma^{(L-1)}...(\sigma^{(1)}(xW_1)W_2, ...)W_{L-1})W_L), \quad (3)$$

where the $\sigma^{(i)}$ represent (possibly non-linear) activation functions and W_i represents the linear operator in i-th layer, respectively. The beauty of the neural network is that it (approximately) characterizes the complicated and infinite continuous function space with a function class of finite parameters. A large network is formed by simple bases, through the generalized linear model $\sigma(xW)$ on input x. Though the approximation capacity of each unit of $\sigma^i(xW_i)$ is limited, the integration is rich. If we represent the sample in a vector x and let σ be some coordinate-wise activation function, such as Relu or Sigmoid, (3) just captures a fully-connected network. There are a huge number of variants such as convolutional network, recurrent network, long-short term memory, auto-decoder and Transformer.

[3] Prior knowledge is necessary in the PAC security definition. One may imagine the case that an adversary has full knowledge of x, where the distribution P is reduced to a single point (with zero entropy). An adversary can always exactly recover x regardless of the ciphertext.

At a high level, we may force these more complicated architectures into (3), but the function $\sigma^{(i)}$ and the dimensions of W_i may require more complex restrictions. We can always view the neural network as a large generalized linear model and the following identity holds for any invertible matrix W:

$$\begin{aligned}\mathcal{N}(x) &= \sigma^{(L)}(\sigma^{(L-1)}(...(\sigma^{(1)}(xW_1)W_2...)W_{L-1})W_L) \\ &= \sigma^{(L)}(\sigma^{(L-1)}(...(\sigma^{(1)}(xW\cdot W^{-1}W_1)W_2...)W_{L-1})W_L) \\ &= \mathcal{N}'(xW).\end{aligned} \tag{4}$$

Taking T simply as a (possibly randomly selected) invertible transformation as $T(x) = xW$, (4) shows that if the ground truth model f of sample x can be well approximated by some $\mathcal{N}(x)$, then the optimal model $f \circ T^{-1}$ for transformed data $T(x)$ defined above is still within the expressibility of the same network architecture. The idea presented above is very generic: Once we rewrite a network such that its *lowest layer*, i.e., layer closest to the input, can be rewritten as a simple generalized linear form, an appropriate linear transformation can be easily constructed. In [47], a simple transform for a fully-connected network was presented with an associated privacy guarantee relating the number of exposed transformed samples to the entropy of the input data.

2.4 A Framework of PAC Security Analysis

In this subsection, we overview the methodology proposed in Dauntless to analyze PAC security of a transformation. To estimate a lower bound of the sample complexity for an (ϵ, δ) PAC approximation, we consider how much information (in bits) is at least required for approximation with this performance and how much information (in bits) is provided by each transformed sample under given prior knowledge. To be formal, for any inference algorithm Adv dependent on m observations $\mathcal{S} = T(x)^m \sim P^m$ in (2), we consider the mutual information $I(T^{-1}; g)$. Since the algorithm g can be viewed as a post processing on $\mathcal{S}$, thus we have

$$\begin{aligned}\mathcal{I}(T^{-1}; g) &\le \mathcal{I}(T^{-1}; S) = \mathcal{H}(T(x)^m) - \mathcal{H}(T(x)^m|T^{-1}) \\ &\le m\mathcal{H}(T(x)) - m\mathcal{H}(T(x)|T^{-1}) = m\mathcal{I}(T(x); T^{-1}).\end{aligned} \tag{5}$$

Here, the last inequality is due to the use of the fact that the joint distribution $\mathcal{H}(a, b) \ge H(a) + H(b)$, and $T(x_1)$ and $T(x_2)$ are independent conditional on T, which is bijective to T^{-1}. Therefore, we have a natural lower bound on the sample complexity m as

$$m \ge \frac{\mathcal{I}(T^{-1}; Adv)}{\mathcal{I}(T^{-1}; T(x))},$$

for x distributed as P. The denominator $\mathcal{I}(T^{-1}; T(x))$ is relatively easy to handle where

$$\mathcal{I}(T^{-1}; T(x)) = \mathcal{I}(T; T(x)) = \mathcal{H}(T(x)) - \mathcal{H}(T(x)|T).$$

Here, we still exploit the fact that T^{-1} is bijective to T. The trickier part is the estimation of the numerator. The idea in Dauntless is to split the domain of T^{-1} into several subsets such that any two T_1^{-1} and T_2^{-1} selected from different subsets sufficiently differ from each other, where

$$\Pr_{x\sim P,T}(\|T_1^{-1}(T(x)) - T_2^{-1}(T(x))\| > 2\epsilon) > 2\delta.$$

Such a construction allows the application of Fano's inequality to produce a lower bound.

2.5 Contrast with Encryption Approach

Limited prior knowledge and the random transformation produce two natural challenges that an adversary must address before making any meaningful inference in practice, even in the simplest case where the transformation is a random invertible linear operator, characterized by a $d \times d$ invertible matrix. It may seem that inverting the transformation can be as simple as solving a linear system, where d known plain samples are sufficient.

First, the private transformation setup is *not* equivalent to that of a cryptographic encryption, where an adversary may arbitrarily select polynomial many plaintexts and request corresponding ciphertexts to guess the key. In the private learning scenario, the owner *"encrypts" her own data*. Depending on the confidence of the secrecy of the data, the owner adjusts the publishing strategy. For example, to strengthen the entropy of each sample, one technique described in Dauntless [47] is to use a data augmentation technique *mixup* [51][4], so even when public images are used for training, they are only used after mixing with a private image that is unknown to the adversary. Further, one can always incorporate some other randomness, such as random cropping and erasing [43], [52], commonly used data augmentation techniques, or the data set can be augmented with labeled noise as described in Section 6, to amplify the private samples' entropy. Thus, with limited prior knowledge, breaking the linear system can be hard, as captured by the PAC security theory.

The second and more straightforward challenge is the unknown correspondence. Transformed samples are randomly shuffled. Even if the adversary has partial knowledge on a subset of the secret data set, due to the random transformation, determining the correspondence is a hard problem, known as (noisy) random linear observation with unknown permutation (unlabeled sensing) [23, 34, 44]. Even in the noiseless case, which corresponds to that of an adversary having full knowledge on the data set, determining the Maximal Likelihood Estimation (MLE) of the permutation can be NP-hard [34, 35].

As a summary, our sample complexity lower bounds are based on the uncertainty or entropy of the private samples amplifying the number of exposed transformed samples required for recovery. Indeed, the PAC security of Section 3 captures a much stronger adversary given access to perfect correspondences between each plain and transformed sample, while in practice recovery is harder. We defer a comprehensive PAC security study, which takes both permutation and random transformation into account, to future work.

3 PRIVATE TRANSFORMS AND PAC PROOFS

We review the transform of [47] in Section 3.1 and present a more utility-friendly variant and prove a more precisely quantified sample bound based on PAC security. We review Transformer networks in Section 3.2, and describe a new private transform for Transformers with an associated privacy guarantee in Section 3.3. We review Convolutional Neural Networks (CNNs) in Section 3.4, and describe a

[4] *Mixup* achieves significant success in semi-supervised learning, e.g., [3], and has been used to improve utility, e.g., [46].

new private transform for CNNs with an associated privacy guarantee in Section 3.5. Finally, in Section 3.6, we discuss application to Transfer Learning.

3.1 Fully Connected Network Transform and Privacy Guarantees

Similar to Dauntless, we first consider the transformation T to be a single fully-connected layer with a random weight matrix W, i.e., $T(x) = \sigma(xW)$. Without loss of generality, we assume $T : \mathbb{R}^d \rightarrow \mathbb{Q}^d$, where $\mathbb{Q}$ is some finite set, capturing a quantification with limited precision. In Dauntless, the authors select the random matrix W to be an *inverse* of a uniform matrix and point out that a random matrix has a high probability of being ill-conditioned where the least singular value is small. Training over such transformed data becomes unstable and therefore the matrix requires additional processing. In this paper, we propose a new and more generic framework to achieve PAC security.

First, we consider a more utility-friendly transformation, where the weight matrix W is selected to be a random uniform matrix. To give a simple example for theoretical analysis below, we consider that each entry of W is randomly selected from $\{-1, 0, 1\}$. It is noted that the expectation of each entry equals 0 and based on the generalization of Johnson–Lindenstrauss Lemma [29], a random matrix of zero mean has the potential to preserve the transformed sample pairwise distance. Such kind of transformation which preserves certain sample space geometry will ease the following training procedure.

Second, we provide the sketch of the new PAC security proof which relies on the Hanson–Wright inequality to give a tighter sample bound for a more practical security budget. Recall that what (ϵ, δ) PAC approximation says is that with probability at least $(1 - \delta)$, the adversary can recover the secret input with an approximation error smaller than ϵ. The transformation matrix $W \in \mathbb{R}^{d \times d}$ is randomly selected from a set of 3^{d^2} elements. Assume that an adversary estimator Adv can successfully recover the secret input x transformed by W_0, where $\Pr_{x \sim P}(\|Adv(\sigma(xW_0)) - x\| < \epsilon) \geq 1 - \delta$. Then, we want to check how many other matrices W exist such that Adv can also successfully invert $\sigma(xW)$. We set out to show that the true transformation matrix is hard to distinguish amongst exponentially many possible candidates given the transformed samples, and any guess of the true inverse is limited to only handling a negligible fraction of transformations simultaneously.

To support the above goal and avoid tedious discussions in the theoretical analysis, we add two additional restrictions on the generation of W from $\{0, \pm 1\}^{d \times d}$. If W generated is non-invertible, we consider the following operator. Let the singular value decomposition (SVD) of W be $W = USV$ and we take $\hat{S} = S + \lambda \cdot \boldsymbol{I}_d$, for some $\lambda > 0$. In other words, we add a positive constant λ uniformly to the original singular values of S and clearly $\hat{W}$ is then invertible. Moreover, it is noted that $\|\hat{W} - W\|_F = \|\lambda UV\|_F = \lambda\sqrt{d}$, since U and V are both unitary matrices. Since λ is arbitrary, we will simply let $\lambda \rightarrow 0$ and the perturbation on W is negligible. If W generated is non-invertible, we will take $\hat{W}$ instead. Thus, in the following, we will simply take W as an invertible matrix. Further, we will filter out all W from $\{0, \pm 1\}^{d \times d}$ if $\|W\| > c\sqrt{d}$ for some c.

THEOREM 1. *Assume the distribution P (prior knowledge) of x to be a multivariate Gaussian distribution, $\mathcal{N}(0, \tau \boldsymbol{I}_d)$, and the weight matrix W of the transformation $T = \sigma(xW)$ [5] to be randomly selected from $\{0, \pm 1\}$ such that $\|W\| \leq c\sqrt{d}$, then such a mechanism T satisfies (ϵ, δ)-PAC security if the number m of samples exposed satisfies*

$$m \leq \frac{\log\left((1 - e^{-log(3/2)cd})3^{d^2}\right) - \log\left(\binom{d^2}{\beta d^2} \times 2^{\beta d^2}\right)}{\mathcal{I}(T^{-1}; \sigma(xW))} \tag{6}$$

where $\mathcal{I}(T^{-1}; \sigma(xW))$, which equals $\mathcal{H}(\sigma(xW)) - \mathcal{H}(\sigma(xW)|W)$, can be upper bounded by

$$\mathcal{I}(T^{-1}; \sigma(xW)) \leq d \sum_{o \in Q} \Big(- p_1(\sigma^{-1}(o)) \log(p_1(\sigma^{-1}(o)))\Big) + o(1),$$

Here, let $Q_1(\cdot)$ be the probability density function (pdf) of $\mathcal{N}(0, \tau\|v\|)$ where $\|v\|^2 \sim B(d, 2/3)$, a binomial distribution, and $p_1(\sigma^{-1}(o)) = \int_{z \in \sigma^{-1}(o)} Q_1(z)dz$. Additionally, ϵ and δ satisfy the following,

$$\epsilon = \frac{\sqrt{\tau^2 \beta d^2 - t}}{2c\sqrt{d}}, \text{ and } \delta \leq \frac{1 - e^{-\frac{t}{8c^2 d\tau^2}}}{2}, \tag{7}$$

with free parameters $c > 1$, $t > 0$ and $\beta > 0$.

The proof of the above theorem is in Appendix A. Asymptotically, Theorem 1 tells that after exposing $O(d)$ transformed samples, we have resistance to PAC inference with estimation error $\epsilon = \Theta(\tau\sqrt{d})$ when the original secret samples are uncertain to the adversary, captured by a Gaussian. A more heavy-tailed distribution (more uncertainty) will produce a larger ϵ.

Remark 1: The framework presented here works for all kinds of sub-Gaussian random matrices W where an asymptotically similar security guarantee can be provided. For example, one can change the generation of matrix W to be a random binary matrix $\{0, 1\}^{d \times d}$. It is noted that, with a random binary matrix, even if we assume that the adversary has the full knowledge of the secret data, determining the encoding matrix W will be reduced to a (multidimensional) Subset Sum Problem [17]. In the worst case, it can be NP-hard, though information theoretically, the produced sample bound has only a constant difference from what is provided in Theorem 1.

We give a concrete example of the above sample bound. When we select $\tau = 1$, $\beta = 0.2$, $c = 1.1$ and $t = 32c^2 d$, it produces (ϵ, δ) at least $(0.2\sqrt{d}, 0.49)$ for $d > 3000$. For the sample bound, we approximate the $\mathcal{I}(T^{-1}; \sigma(xW))$ by $\sum_{i=1}^{d} \mathcal{I}(T^{-1}; \sigma(xW)(i))$, where $\sigma(xW)(i)$ denotes the i-th coordinate of $\sigma(xW)$. If $\sigma(\cdot)$ is a quantification function which only keeps the first three digits after the decimal point, then $\mathcal{I}(T^{-1}; \sigma(xW)) \approx 0.0138d$ and provides the (ϵ, δ) guarantee when at most $m = 31.5d$ samples are exposed.

3.2 Transformer Network

Transformers are a relatively modern network architecture, which outperforms many existing architectures based on recurrent or convolutional layers in many Natural Language Processing (NLP) tasks [45] [12]. Recently, image Transformers [14] have become competitive with CNNs in visual recognition tasks.

[5] When σ is non-invertible, we simply assume $T^{-1} = \sigma^{-1}(\cdot)W^{-1}$ and σ^{-1} is an oracle such that $\sigma^{-1}(\sigma(xW)) = x \cdot W$.

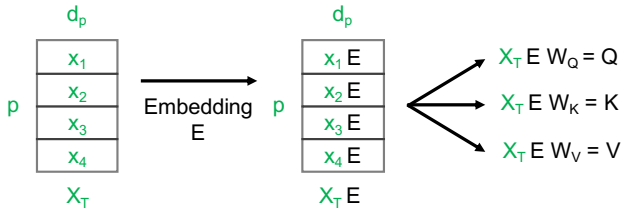


Figure 1: First part of a Transformer with $p = 4$ patches

Without loss of generality, assume each training sample x is formed by p segments, $\boldsymbol{x} = [\boldsymbol{x}_i, \boldsymbol{x}_2, ..., \boldsymbol{x}_p]$, where each $\boldsymbol{x}_i \in \mathbb{R}^{d_p}$ can be a patch of an image or a word in a sentence. Put $\boldsymbol{x}$ into a matrix form $X_T \in \mathbb{R}^{p \times d_p}$, where the i-th row corresponds to $\boldsymbol{x}_i$. The first step in the Transformers we consider is the multiplication of each patch by a learnable $d_p \times d_p$ embedding matrix E as shown in Fig. 1 to produce an embedded $p \times d_p$ matrix $X_T E$.

The fundamental method applied in a Transformer to processing the data is Self Attention to measure the relevance amongst different $\boldsymbol{x}_i$. In a Self Attention model, there are three weight matrices $W_q, W_k, W_v \in \mathbb{R}^{d_p \times d'}$ to be trained (see Fig. 1), corresponding to the Query, Key and Value, defined below. Then, we call $Q = X_T W_q$, $K = X_T W_k$ and $V = X_T W_v$ as Query, Key and Value vectors, respectively, each of dimension $p \times d'$. The output of the Self Attention is defined as

$$softmax(\frac{Q * K^T}{\sqrt{d'}}) \cdot V. \tag{8}$$

Here, $*$ can be either the dot product or matrix multiplication.

3.3 Private Transform for a Transformer Network

Now, we will describe an appropriate transformation T. Perform an independent linear transformation on each row of X_T, which corresponds to each segment of $\boldsymbol{x}$, say

$$X_T \rightarrow \begin{bmatrix} \boldsymbol{x}_1 \tilde{W}_1 \\ \boldsymbol{x}_2 \tilde{W}_2 \\ ... \\ \boldsymbol{x}_p \tilde{W}_p \end{bmatrix} \tag{9}$$

where each $\tilde{W}_i$ is a $d_p \times d_p$ random matrix. Therefore, each patch is being multiplied by different private randomness as shown in Fig. 2. To preserve the representation capacity of a Transformer, we add an E_i learnable fully connected layer for each $x_i \tilde{W}_i$ and feed the result to the Transformer as illustrated in Figure 2. Given that we have added this learnable *matching* layer, we remove the E embedding matrix. Essentially, the E matrix of Figure 1 is replaced by p E_i matrices in Figure 2.

Consider the original sample to be a vector of length $d = p \cdot d_p$. In the case of the fully connected network (see Section 3.1) recall that we have a $d \times d$ random matrix W as the private transform. Here, we have p $\tilde{W}_i$ random matrices, each of dimension $d_p \times d_p$, as the collective private transform, meaning there is a factor of p less randomness, which affects the sample bound by the corresponding factor.

THEOREM 2. *Assume the distribution $\mathcal{P}$ (prior knowledge) of x to be a multivariate Gaussian distribution, $\mathcal{N}(0, \tau \boldsymbol{I}_d)$, and the transformation weight matrix $\tilde{W}_i$ for each patch to be independently*

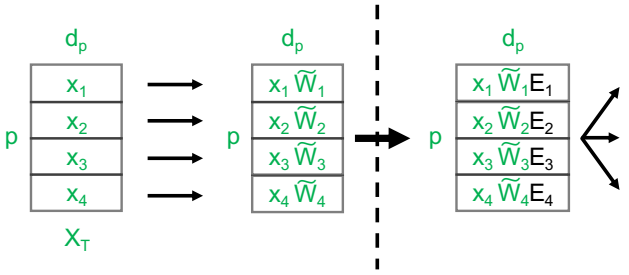


Figure 2: A Transformer with Additional Matching Layer

and randomly selected from $\{0, \pm 1\}^{d_p \times d_p}$ such that $\|\tilde{W}_i\| \leq c\sqrt{d}$, then such a mechanism T satisfies (ϵ, δ)-PAC security with (ϵ, δ) described in (7), if the number m of samples exposed satisfies

$$m \leq \frac{p \log\left((1 - e^{-\log(3/2) c\sqrt{p} d_p}) \cdot 3^{d_p^2}\right) - \log\left(\binom{p \cdot d_p^2}{\beta d^2} \times 2^{\beta d^2}\right)}{\mathcal{I}(T^{-1}; \sigma(xW))}, \tag{10}$$

where $\mathcal{I}(T^{-1}; \sigma(xW)) = p\big(\mathcal{H}(\sigma(\tilde{x}\tilde{W})) - \mathcal{H}(\sigma(\tilde{x}\tilde{W})|\tilde{W})\big)$, $\tilde{x} \sim \mathcal{N}(0, \boldsymbol{I}_{d_p})$ and $\tilde{W}$ corresponds to the random $d_p \times d_p$ weight matrix generated for each patch. Here, $c > 1/\sqrt{p}$, $\beta > 0$ and $t > 0$ are free parameters.

The proof of Theorem 2 is in Appendix B. It is noted that for any parameter selection (c, β, t) in (7), the scaled $(c, \beta/p, t/p)$ produces the same (ϵ, δ). Therefore, under the same setup, the sample complexity in Theorem 2 will be $O(p)$ smaller than that of Theorem 1.

3.4 Convolutional Neural Network (CNN)

In a CNN, a convolution is a linear multiplication of a set of weights with the input data, typically an image. The multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel. The $k \times k$ kernel is typically much smaller than the $\sqrt{d} \times \sqrt{d}$ input data. The type of multiplication applied between a kernel-sized patch of the input and the kernel is a *dot product* or scalar product.

Using a kernel smaller than the input allows the same kernel (set of weights) to be multiplied by the input array multiple times at different positions on the input. Elaborating, the filter is applied systematically to each (possibly overlapping) kernel-sized patch of the input data, left to right, top to bottom. The amount of overlap is controlled by a stride parameter, that can vary from 1 to k.

The first layer of a CNN is typically a convolutional layer, which consists of many learnable kernels. During the forward pass, we convolve each kernel across the width and height of the input image as described above. During training, the network will learn kernels that activate when some type of visual feature such as an edge of some orientation is seen. Each kernel in the convolutional layer produces a separate 2-dimensional activation map that is processed by subsequent layers.

The key observation is that these kernels perform *local computations*, and the private matrix transform for the fully connected network is a global transform that matches a fully connected layer. Not surprisingly, transforming images in such a fashion results in significantly degraded utility in CNN-based object detection.

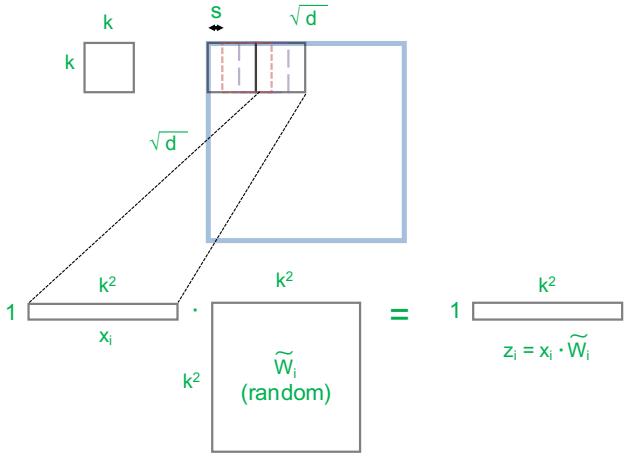


Figure 3: Private Transform for CNN

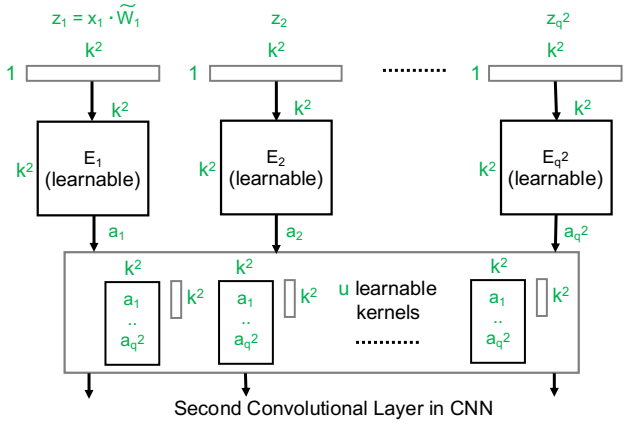


Figure 4: CNN with Additional Matching Layer and Modified First Layer

3.5 Private Transform for a CNN

Figure 3 illustrates the private transform tailored for CNNs. We pretend that we are convolving the image with a $k \times k$ kernel, and *each* $k \times k$ window of the input image is converted to a $1 \times k^2$ x_i vector, similar to the patches in a Transformer Network described in Section 3.2. The only difference is that x_i may be overlapped. Each x_i is privately transformed to be multiplied by a different random $k^2 \times k^2$ $\tilde{W}_i$ matrix, which maintains locality within the window. Depending on the stride s, different numbers of $1 \times k^2$ vectors are generated, one for each position of the pretend kernel. A total of $q^2 = (\frac{\sqrt{d}-k}{s}+1)^2$ many transformed $z_i = x_i\tilde{W}_i$ vectors are generated.

Now, we describe a modified CNN architecture suitable for training over the transformed data. Figure 4 shows the additional matching layer that we require in front of the CNN (similar to the Transformer case) for efficient learning. A total of $(\frac{\sqrt{d}-k}{s}+1)^2$ fully-connected layers of dimension $k^2 \times k^2$ are placed before the CNN. In addition, since we have already generated the convolution windows over the input in the transform, we modify the first convolutional layer of the CNN to only perform the multiplication of the provided input by a number of learnable weight kernels as shown in Figure 4; we assume the original CNN has u learnable kernels in its first layer. The second and subsequent layers in the CNN are unchanged.

Indeed, the security guarantee for the above private transformation for CNN is almost the same as what is described in Theorem 2 even when the patches are (possibly) overlapped. The details can be found in Appendix C.

3.6 Transfer Learning

Through the above three examples, we have introduced the key idea behind the conservative transformation framework. We now briefly discuss transfer learning with private transformation. Transfer learning is a broad machine learning concept, where to address a new task, one may start by reusing an earlier model trained for a similar task. Transfer learning has received great success. For example in a Transformer network, if we initialize the weights by those trained on the public Imagenet data set, the classification accuracy over CIFAR-10 can be improved by at least 20% [14]; a small and easier image classification task may enjoy the experience gained from solving a larger and more challenging problem.

Usually, in deep learning, the bottom layers extract more fundamental and general features. To incorporate the private transformation idea into the context of transfer learning, for a given pretrained model, a data owner may split and take the first several layers of the network out as a public fixed function while uniformly transforming the secret samples through it. Then, depending on the particular subsequent network architecture, the owner can apply a proper private transformation on the processed data. For example, consider a CNN, which is formed by multiple convolutional layers in the beginning with fully-connected layers at the end. Given pre-trained parameters, the owner can first let secret samples pass through the first convolutional layers, and then apply the private transformation shown in Section 3.1. Then, the user can send those transformed samples with a request to train the remaining layers.

The above idea also captures the data embedding scenario. In many NLP tasks, one has to first select an appropriate pre-trained model such as BERT [13], to efficiently embed the input, for example a sentence, to dense vectors, which are then fed to train a model. In such a case, depending on the specific network used, the private transformation can be naturally applied on the embedded data rather than the original data.

4 TASK AUGMENTATION

The balance between generalization and empirical loss minimization is one of the most challenging problems in machine learning. Such a tradeoff has been extensively studied in existing works, e.g., [5, 21, 50]. In practice, there are two common approaches to improve generalization but reduce memorization. One is data augmentation, where training is implemented over similar but more fuzzy data. Many useful techniques have been discovered such as mixup [51], random cropping [43] and erasing [52]. The other is on the optimization side such as adding regularization [30] or more robust loss functions [1, 18] resistant to outliers. All of these improvements over either the data representation or model training apply even for transformed data. Here, we consider this problem from a different perspective.

The key idea behind Task Augmentation (see Algorithm 1) is that we aim to train a model for a more general task, where the original learning task may become a subproblem. Suppose one is working on a c-classification problem with n training samples in a form (x_i, y_i), where the feature $x_i \in \mathbb{R}^d$ while label $y_i \in \mathbb{R}^c$ is a one-hot vector. Now, imagine there is another data set corresponding to c'-classification task, whose samples are in a form (x'_i, y'_i), where $x'_i \in \mathbb{R}^{d'}$ with one-hot vector label $y'_i \in \mathbb{R}^{c'}$. In general, the c'-classification task is not necessarily related to the original c-classification. Then, we define a new $c \times c'$-classification by considering the Cartesian product of any pairs $\{(x_i, y_i), (x_j, y_j)\}$ in a form $\big((x_i|x'_j), (y_i|y'_j)\big)$. The composite feature domain is now within the $(d + d')$-dimensional space while the label domain also increases corresponding to $c \times c'$ classes.

Algorithm 1 Task Augmentation

Input

(a) Main training data set of n samples $\big((x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\big)$ with feature $x_i \in \mathbb{R}^d$ and one-hot label vector $y_i \in \mathbb{R}^c$ for c classes.

(b) A (separate) auxiliary training data set of n' samples $\big((x'_1, y'_1), (x'_2, y'_2), ..., (x'_n, y'_n)\big)$ with feature $x'_i \in \mathbb{R}^{d'}$ and one-hot label vector y'_i for c' classes.

(c) A query x asking for inference.

Phase 1: Data Generation and Training

for $i = 1 : m$ **do**
  Randomly select $j_i \in [1 : n]$ and $l_i \in [1 : n']$.
  Generate a composite sample $(\tilde{x}_i, \tilde{y}_i)$ in a Cartesian product, where $\tilde{x}_i = (x_{j_i}|x'_{j_i})$ and $\tilde{y}_i = (y_{j_i}|y'_{j_i})$.
end for
Train a model f on $(\tilde{x}_{[1:m]}, \tilde{y}_{[1:m]})$.

Phase 2: Inference

Choose $m' \in [1 : n']$
for $j = 1 : m'$ **do**
  Evaluate model f on (x, x'_j) and let $\bar{y}_j = f\big((x|x'_j)\big)$.
end for
Calculate $\bar{y} = \sum_{j=1}^{m'} \bar{y}_j$.
Output $i^* = \arg\max_{i \in [1:c]} \bar{y}_j(i)$ as the inference result.

We have three remarks on Task Augmentation. First, though the learning task becomes more complicated with a feature dimension increase, random regrouping of sample pairs from two data sets also produces a larger sample pool. Second, during inference, instead of a single evaluation, the prediction benefits from a more comprehensive test with multiple different composite samples. Ideally, one may apply the entire auxiliary training data to enhance the inference. From our experiments, usually $10c'$, i.e., around 10 auxiliary samples per auxiliary class, is enough to guarantee good performance. From the above protocol, it is clear that Task Augmentation is an independent framework, which does not require any specific restrictions on either the sample representation or the training mechanism selection. Third, we note that Task Augmentation is easily generalized to more than one auxiliary data set.

Implementation: We now describe detailed implementations of Task Augmentation in different neural networks.

The samples in the Cartesian product can be straightforwardly handled by a fully-connected network, where one can simply scale the number of weights in each neuron to match the input size.

As for image processing in CNNs, there are two natural ways to incorporate Task Augmentation. One is to put one image just below the other to produce a new taller picture. For example, two 32×32 images can form a 64×32 image. The other strategy is to put the two images into two channels, where we feed a $32 \times 32 \times 2$ image to the network. Correspondingly, if originally kernels of size $k \times k$ are applied in the first convolution layer, then the kernel size doubles to be $k \times k \times 2$ to handle the convolution over the composite input. Depending on the particular network architecture, one may select the proper strategy.

The implementation of Task Augmentation in Transformer networks is a combination of that in fully-connected networks and CNNs. Recall that the first layer of a Transformer is a fully connected network while the input is not a vector but a patch matrix, similar to a CNN. So for two inputs $x_1 = (x_{11}, x_{12}, ..., x_{1p})$ and $x_2 = (x_{21}, x_{22}, ..., x_{2p})$, we merge them into

$$\tilde{x} = \big((x_{11}|x_{21}), (x_{12}|x_{22}), ..., (x_{1p}|x_{2p})\big),$$

where each patch size doubles and correspondingly the number of weights to learn in the first fully-connected layer doubles.

5 PRIVATE COLLABORATIVE LEARNING

Now, we can proceed to describe the private collaborative learning protocol. Imagine that there are two data set owners and each holds private samples. We consider the extreme case that both users do not trust each other but aim to privately train a model utilizing the samples from both via an untrusted server. To preserve the local data privacy, a natural generalization of Dauntless protocol (see Section 2) is for each owner to independently transform her own data and send to the server; on the other side, the server trains over the aggregated transformed samples.

However, such trivial extensions encounter a utility dilemma. Imagine that the two owners have samples of a common class, where through a straightforward data sharing or collaborative training they may easily produce a better model. However, after independent transformation, if we still label samples the same, the difficulty of learning increases significantly since similar samples after independent transformations may differ dramatically. Similarly, if we label them differently, it is equivalent to addressing the two owners' respective classification tasks independently, and the performance is no better than that of training individually. It may seem that independently transformed data only complicates the training procedure without benefiting from it.

Another direction is Multi-Party Computation (MPC), where through cryptographic primitives data owners may collaboratively produce and share a same transformation, without either one knowing the transformation. However, in such a setup, even if the resultant trained model is shared by all owners, each time a given owner wants to utilize the model, the implementation requires the assistance of the other one to first transform the input so the model can be evaluated, which can be costly. Furthermore, the privacy budget of the transformation is always limited, therefore the number of predictions is bounded.

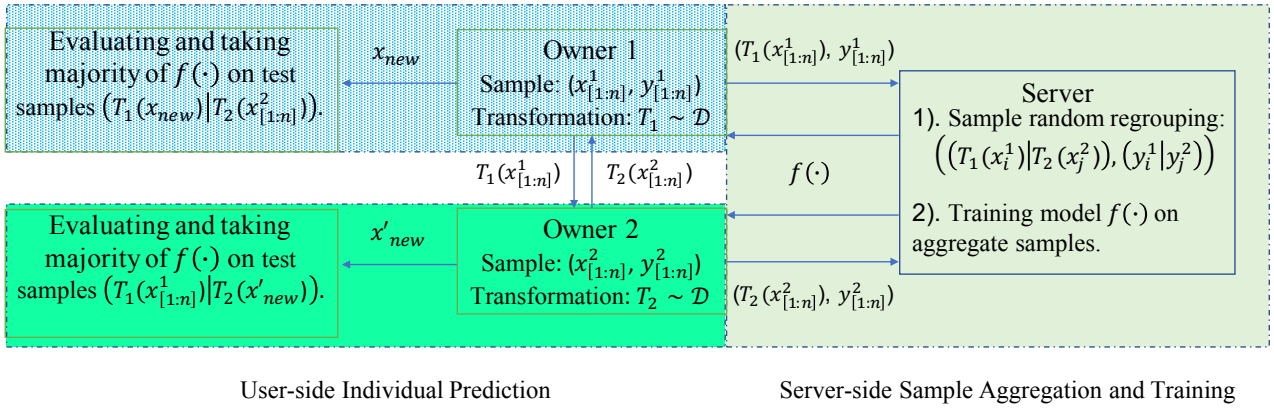


Figure 5: The Task Augmentation Framework for Private Collaborative Learning

Task Augmentation addresses this challenge elegantly. It is noted that even if the respective transformations are independently selected but distributed identically, in a global view, identically distributed samples after such transformations are still distributed the same. Task Augmentation based collaborative learning for multiple data owners is presented in Figure 5. In general, assume that there are K owners and without loss of generality, assume each owner has n samples. First, for the data transformation procedure, each owner independently generates a transformation $T_i, i \in [1:K]$, from some fixed distribution $\mathcal{D}$. Transformed data $(T_i(x^i_{[1:n]}), y^i_{[1:n]})$ is then sent to the server. It is worth emphasizing that each owner can independently select her own labeling strategy of $y^i_{[1:n]}$ [6]. Second, on the server, Kn samples are randomly regrouped as $\left((x^1_{i_1}, y^1_{i_1}), ..., (x^K_{i_K}, y^K_{i_K})\right)$, where at most n^K different combinations can be produced. The server then takes the corresponding K-Cartesian product on both the K features and labels as aggregate samples, $\left((x^1_{i_1}|x^2_{i_2}|...|x^K_{i_K}), (y^1_{i_1}|y^2_{i_2}|...|y^K_{i_K})\right)$, and trains a model $f(\cdot)$ over them. Model $f(\cdot)$ is then sent back to all the owners. Third, for the application of $f(\cdot)$, each owner publishes at least one, a few, or all the transformed $T_i(x^i_{[1:n]})$ features to other owners. There are no additional security assumptions required on the other owners and the server, since the transformed samples $(T_i(x^i_{[1:n]}), y_i)$ have been exposed to the server in the first step, and this additional sharing of transformed features amongst owners will not cause any further data privacy loss. Now, when owner i wants to apply $f(\cdot)$ on a newly incoming x_{new} for prediction, with $T_i(x^i_{[1:n]})$ at hand, at most n^{K-1} many virtual test samples can be constructed as $\left(T_1(x_{j_1})|....|T_i(x_{new})|...|T_K(x_{j_K})\right)$ for $j_l \in [1:n]$. After embedding the x_{new} sample into the composite domain and applying $f(\cdot)$ on some number of virtual samples, owner i may take the majority on the i-th segment as the outputs of $f(\cdot)$. Ideally, a well-trained $f(\cdot)$ will produce

$$f\left(T_1(x_{j_1})|....|T_i(x_{new})|...|T_K(x_{j_K})\right) = (y^1_{j_1}|...|y_i|...|y^K_{j_K}),$$

where y_i is the true label of x_{new}. Clearly, in the above construction, owner i does not necessarily need the knowledge of other $y^l_{j_l}$, $l \neq i$; owner i only need focus on the i-th segment, and a good estimation can be the majority of all evaluations. Furthermore, during the individual prediction, the implementation does *not* require the participation of other users as in the MPC case; with the assistance of at least one published transformed sample from each owner, one can already enjoy the benefits of the collaboratively learned model.

[6] Under the Cartesian product, the owners' respective label space is independent. Thus, every owner only needs to guarantee that the labels are orthogonal, such as one-hot vectors, for her different classes.

6 AUGMENTATION WITH LABELED NOISE

In Section 5, we described a private collaborative learning protocol where the various tasks are typically addressing similar learning problems. Indeed, Task Augmentation strengthens robustness in general, and when no private transformation is applied, the resulting utility will be at least the worst utility of the multiple tasks addressed individually. However, similarity of tasks is not a necessary condition to apply Task Augmentation. In this section, we explore using Task Augmentation in a single data owner setting for *privacy amplification*, where multiple tasks address very different problems. In particular, we generate random data sets, which are easy to classify such as (linearly) separable sets, and utilize them for privacy amplification.

As before, the c-classification task corresponds to the owner's training task. Consider synthesizing linearly separable noise, and adding this *labeled noise* c'-classification task to the c-classification data set by defining a new $c \times c'$-classification problem as described in Section 4. The key difference here is that the private transform is applied *after* Task Augmentation, as opposed to before Task Augmentation in the collaborative multiple owner case of Section 5. We can do this because the real data set and the learnable noise data set are owned by the same entity. Applying the private transform after Task Augmentation effectively obfuscates the original data set over and beyond just applying the transform since noise is mixed into the original data.

We now describe how to synthesize learnable noise. We consider two separate sets. Let (x^e, y^e) be a synthesized (noise) sample, which is binary-labeled and $y^e \in \{\pm 1\}$. When $y^e = 1$, x^e is randomly generated such that each coordinate $x^e(j) \sim \mathcal{N}(\mu, \tau)$, $j = 1, 2, ..., d$, i.e., it follows a Gaussian distribution with mean μ and variance τ. Similarly, when $y^e = -1$, let $x^e \sim \mathcal{N}(-\mu \cdot \mathbf{1}, \tau \cdot \boldsymbol{I}_d)$, where $\mathbf{1} =$

$(1, 1, ..., 1)$. Clearly, when $\mu \gg \tau$, we simply construct two separate sets as the synthesized data set. [7]

Consider the following composite sample form $\big((x|x^e), (y|y^e)\big)$, where $x \in \mathbb{R}^d$ is the original sample and (x^e, y^e) is a binary-labeled sample synthesized as described above. We apply the private transform to composite samples, $(x|x^e) \cdot W$. This corresponds to multiplying by a matrix (or matrices) of appropriate dimension depending on the network that we will train on (see Section 4). One can view Task Augmentation with noise as adding randomness and entropy to the private data.

The above framework can also be straightforwardly applied to produce (L)DP guarantees even without the private transform by simply releasing the $\big(x + x^e, (y|y^e)\big)$ data set. It is noted that the privacy guarantee is characterized by the variance of the noise, and the selected mean parameter μ does not cause any additional privacy loss – the sensitivity stays the same with a uniform shift. Thus, the separable noise $\mathcal{N}(\pm\mu, \tau \boldsymbol{I}_d)$ produces the same (L)DP guarantee as a regular Gaussian mechanism that adds $\mathcal{N}(\boldsymbol{0}, \tau \boldsymbol{I}_d)$. In Section 7.2, we will show how significantly Task Augmentation with labeled noise can improve classification accuracy in comparison to conventional additive noise.

MNIST Non-Private Training			
Network	# Samples	Epoch	Accuracy(%)
FCN	1,000	100	87.7
FCN	2,000	100	91.8
FCN	4,000	50	94.3
FCN	30,000	15	97.8
FCN	60,000	15	98.2
MNIST Single User Private Training			
Network	# Samples	Epoch	Accuracy(%)
FCN	1,000	100	87.6
FCN	2,000	100	91.4
FCN	4,000	50	93.7
FCN	30,000	15	97.8
FCN	60,000	15	98.2
MNIST 2-Collaborative Private Training			
Network	# Samples	Epoch	Accuracy(%)
FCN	$2 \times 1{,}000$	450	91.3
FCN	$2 \times 2{,}000$	225	93.4
FCN	$2 \times 30{,}000$	25	98.1

Table 1: MNIST with Fully Connected Network

7 EXPERIMENTS

7.1 Prediction

We present results on MNIST handwriting recognition and CIFAR-10 object detection data sets. In each experiment, we compare the accuracy obtained on the test suite for non-private and private (transformed data) training, in single user and collaborative settings, and the training times required. The overheads for transforming the data are negligible and therefore not reported.

Three different networks were used in our experiments:

- Fully Connected Network (FCN): A 7 layer network, formed by 2^{10}, 2^9, 2^8, 2^7, 2^6, and 2^5 neurons, with a regression layer at the end.
- Image Transformer Network (ITN): We use the architecture of Vision Transformer [14] with 16 patches.
- Convolutional Neural Network (CNN): We test Resnet 20 and Resnet 56.

Results for MNIST are presented in Table 1 while those for CIFAR-10 are presented in Tables 2, 3 and 4. Since MNIST is a fairly simple data set, we focus on the implementation of MNIST with FCN only. MNIST experiments were run in Matlab R2020a, and CIFAR-10 experiments were run using PyTorch 1.7.1 on a RTX 3090 GPU.

MNIST Results: The entire MNIST data set contains 60,000 samples of 10-digit handwriting pictures and an additional 10,000 test samples. We vary the data set size used for training. With a smooth transformation via a uniform matrix, in a single user case, the test accuracies are essentially the same as non-private training (see Table 1). The training time is also essentially the same – the number of epochs and the time per epoch (not shown) are essentially the same.

Further, collaborative private learning almost matches the non-private case where two owners simply share the plain data and train a single model. It is also better than the corresponding single user case, e.g., 98.1% for the $2 \times 30{,}000$ case is slightly higher than the 97.8% for the non-private/private 30,000 sample case. In the collaborative scenario, the joint task is augmented to a 100 (10×10) classification and $2\times$ longer training time per epoch is required.

We also tested naive collaborative private training, where without Task Augmentation, the server either simply aggregates both owners' respective *transformed* data and trains a 20-classifier over them; or trains a 10-classifier if the two owners have prior consensus on the labeling so only 10 labels are needed. In both cases, the resultant test accuracy degrades significantly to about 70% (not shown). As explained in Section 5, this will happen for all data sets and networks.

CIFAR-10 Results: We include the implementation with FCN on CIFAR-10 in Table 2, primarily for completeness; FCN does not perform well for object detection. We note that test accuracy obtained by private training approaches that of non-private training (53.5% versus 54.6%) and private collaborative training with 2 users, each with 25,000 samples, improves over non-private single user training with 25,000 samples (52.7% versus 49.9%).

For the Transformer network, we split the images into $p = 16$ patches. The results are included in Table 3. ITNs are better than FCNs at object detection; similar to the FCN case, single-user private training produces test classification accuracy close to non-private training, and collaborative private training slightly improves over non-private/private training with half the data set. We also report GPU time per epoch in Table 3. The increase in training time for private single-user training is because of the additional layer in the network (see Fig. 2). There is an additional increase in collaborative learning for the augmented task with double the sample size.

Finally, in Table 4, we present the results of private classification using CNN. We note that for CIFAR-10, under the private transformation described in Section 3.5, the selections of kernel size k within 3 to 5 associated with a stride s no bigger than k only produces small performance differences. Thus, in Table 4, we simply set $k = s = 4$ and $k = s = 3$ for the private training of the single user case and the collaborative case, respectively. CNNs perform

[7] One can also similarly construct two linearly separable sets, where for some selected (secret) $w \in \mathbb{R}^d$, $(x^e.y^e)$ satisfy $x^e \sim \mathcal{N}(\boldsymbol{0}, \tau \cdot \boldsymbol{I}_d)$ and $y^e = sign(x \cdot w^T)$.

CIFAR-10 Non-Private Training			
Network	# Samples	Epoch	Accuracy(%)
FCN	12,500	30	46.4
FCN	25,000	30	49.9
FCN	50,000	20	54.6
CIFAR-10 Single User Private Training			
Network	# Samples	Epoch	Accuracy(%)
FCN	12,500	30	46.2
FCN	25,000	30	49.7
FCN	50,000	20	53.5
CIFAR-10 2-Collaborative Private Training			
Network	# Samples	Epoch	Accuracy(%)
FCN	$2 \times 12{,}500$	120	47.5
FCN	$2 \times 25{,}000$	100	52.7

Table 2: CIFAR-10 with Fully Connected Network

CIFAR-10 Non-Private Training				
Network	# Samples	Epoch	Accuracy(%)	GPU Time(s)
ITN	25,000	60	67.4	1.5
ITN	50,000	40	74.1	3.0
CIFAR-10 Single User Private Training				
Network	# Samples	Epoch	Accuracy(%)	GPU Time(s)
ITN	25,000	60	65.6	5.4
ITN	50,000	40	72.3	7.0
CIFAR-10 2-Collaborative Private Training				
Network	# Samples	Epoch	Accuracy(%)	GPU Time(s)
ITN	$2 \times 25{,}000$	180	67.8	14.2

Table 3: CIFAR-10 with an Image Transformer Network

CIFAR-10 Non-Private Training				
Network	# Samples	Epoch	Accuracy(%)	GPU Time(s)
CNN Res20	25,000	60	89.5	1.1
CNN Res20	50,000	60	92.4	2.2
CNN Res56	25,000	80	89.9	3.0
CNN Res56	50,000	80	93.2	6.0
CIFAR-10 Single User Private Training				
Network	# Samples	Epoch	Accuracy(%)	GPU Time(s)
CNN Res20	25,000	200	85.0	4.1
CNN Res20	50,000	200	88.9	8.2
CNN Res56	25,000	200	86.1	9.1
CNN Res56	50,000	200	89.8	12.0
CIFAR-10 2-Collaborative Private Training				
Network	# Samples	Epoch	Accuracy(%)	GPU Time(s)
CNN Res20	$2 \times 25{,}000$	300	86.3	24.8
CNN Res56	$2 \times 25{,}000$	300	87.8	28.6

Table 4: CIFAR-10 with Convolutional Neural Network

significantly better than ITNs and FCNs for CIFAR-10. The comparisons between non-private training, private single-user training, and private collaborative training are similar to the ITN case, except that private collaborative training does not improve over non-private with half the data set, due to the high efficacy of the half-sized data set; it does improve over private training with the half-sized data set. We do not believe this to be a fundamental limitation and expect to match non-private training with appropriate training scripts.

CIFAR-10 Training with Regular Gaussian Mechanism			
Task	# Samples	Epoch	Accuracy(%)
FCN	50,000	20	28.0
CNN Resnet20	25,000	100	28.2
CNN Resnet20	50,000	100	33.8
CIFAR-10 Training with Labeled Separable Gaussian			
Task	# Samples	Epoch	Accuracy(%)
FCN	50,000	20	34.2
CNN Resnet20	25,000	100	50.6
CNN Resnet20	50,000	100	53.7

Table 5: Private Learning with Non-labeled and Labeled Noise

7.2 Privacy Amplification Using Labeled Noise

Adding noise to plain data is a straightforward way to enhance privacy, especially in the context of (L)DP, where Laplace and Gaussian mechanisms are the most commonly used approaches. Through well-scaled noise (proportional to the sensitivity of output), desired (L)DP guarantees can be provided [16].

In Table 5, we compare the regular Gaussian Mechanism to labeled noise using Task Augmentation in an LDP setting of CIFAR-10 classification. *No private transform was applied*. The CIFAR-10 dataset is normalized where each attribute is within $[0, 1]$. Under the regular Gaussian Mechanism, we add independent Gaussian noise, $\mathcal{N}(0, 1)$, to each attribute of the selected CIFAR-10 data set. Alternately, we generate separable noise data of the same size as the selected training data, where the attribute of a sample is either independently generated from $\mathcal{N}(3, 1)$ or $\mathcal{N}(-3, 1)$, associated with two distinct labels, as described in Section 6. It is noted that the variances of noise are exactly the same in both cases, therefore an identical privacy guarantee is produced. The only difference is that *noise is viewed as an underlying binary classification task in the latter case*, where we add the noise data to the original data and expand the label accordingly to a 20-classification in a Task Augmentation manner. From Table 5, we see that Task Augmentation with separable noise data significantly outperforms the regular Gaussian mechanism, especially in the CNN case, where the classification accuracy on the test set is improved from 33.8% to 53.7%. When noise gets larger, more significant improvements are obtained.

7.3 Image Recovery Attacks

In the following, we experiment with recovery in settings with differing adversarial prior knowledge and show that a small amount of uncertainty makes image recovery difficult as previewed in Section 2.5. Suppose the adversary has partial knowledge of the data set and correspondence between the transformed samples and private samples. In Fig. 6, we present an example where given 15,000 exposed transformed CIFAR-10 images, the adversary can narrow down each private sample within four candidates, i.e., the adversary knows that each private sample is randomly selected from a corresponding four sample set. We assume that 50% of the correspondences between each transformed sample and candidate sets are known. The attacker first creates a data set using each transformed sample as a feature vector, and the label for the feature is chosen as the average of the four candidates, and then trains a neural network to invert the transformation. The attacker then tries to predict the label (private

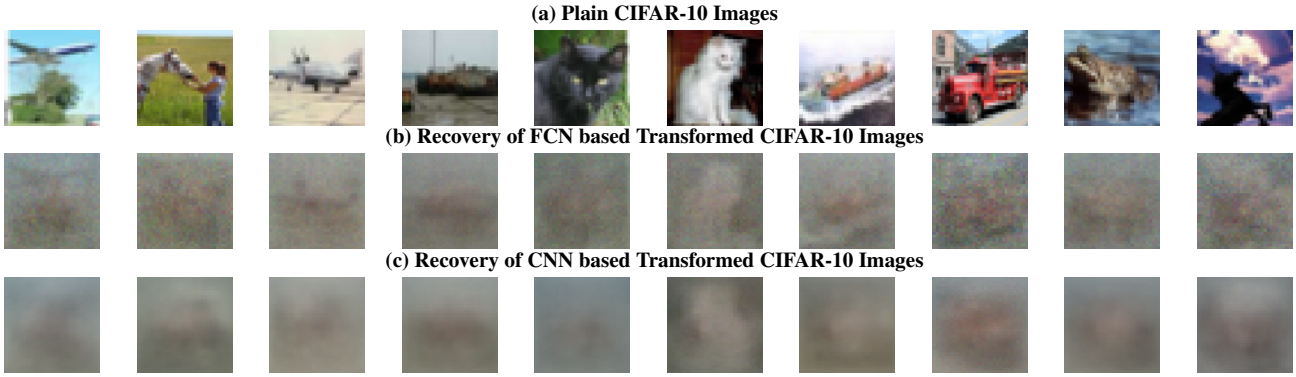


Figure 6: Transformed CIFAR-10 Image Recovery: Prior Knowledge on 50% Correspondences and 4 Candidates Per Plain Sample

sample) for a new feature (transformed sample). In Fig. 6, FCN and CNN transformed data is provided to the attacker, and corresponding recoveries are provided. In particular, in the CNN transformation, we set kernel size $k = 4$ and stride $s = 4$ with further incorporation of random cropping to augment data. The summary is that a small amount of uncertainty impedes recovery significantly. Additional details and experiments are in Appendix D.

8 RELATED WORK

Cryptographic Methods: There has been considerable work applying homomorphic encryption and Garbled circuit techniques for neural network prediction (e.g., [27], [26], [37]). Neural network training is considerably more challenging for cryptographic approaches. Secure multiparty computation (MPC) [19] allows multiple parties to compute a function on their private inputs in such a way that the participants only learn the output of the function and nothing else about each other's inputs. MPC approaches to learning tasks (e.g., [32], [36]) have significant computation overhead and require all parties to collaborate for inference as well as training.

Random Transformation: Random unitary transformations are known to preserve pairwise sample distance and have been used for face recognition applications while preserving privacy (e.g., [28], [33]). Security analysis of the protection schemes is limited to analyzing specific attacks, such as brute force or diversity attacks. The transformations work, i.e., produce good utility, for specific post-processing computations or learning methods such as Support Vector Machines (SVM). More generic random projections with one-time use have been used to achieve secrecy in particular compressive sensing applications [4]. An encryption scheme based on multiplication by a sparse sensing matrix is proposed in [10]. Those transformations and privacy guarantees are only for a single data point and thus the encrypted data cannot be computed on.

Huang et al. proposed a private training protocol, Instahide in [25]. Instahide performs sample-specific sign-flipping unlike the uniform transformation over each sample in the Dauntless framework. Carlini et al. [6] approximates the sample correspondence in Instahide with a similarity graph, and follow-up works [8, 24], which are based on the phase retrieval model, present several attacks on Instahide.

Multi-task Learning: Multi-task learning (MTL) studies training a joint model which solves multiple tasks simultaneously and possibly related tasks, behaving like *hints*, may improve the performance as compared to when tasks are trained separately. For example, in [49] for sentiment analysis, a prediction task on whether an input sentence contains a positive or negative sentiment word is added to the task; in [9] to detect name error, a prediction on whether a name is present in a sentence is added. With a different motivation, Task Augmentation proposed in this paper is a more generic framework, where the point is not to look for efficient *hints* for the main task, but instead to improve generalization by addressing a more complex task and *strengthening the prediction from the more generic model obtained via training.* Task Augmentation can be straightforwardly applied in MTL and we also believe the advanced network architectures [39] proposed in MTL may benefit the implementation of Task Augmentation, which we will explore in our future work.

9 CONCLUSION

We have presented private learning and private collaborative learning strategies with information-theoretic security properties. Using the framework proposed in [47], we have given private data transforms and associated theorems that significantly extend the framework, and are applicable to Transformer networks and CNNs. Importantly, we have presented a Task Augmentation approach that elegantly generalizes the transformation based approach to collaborative learning, without requiring trust between users or in the server. Collaborative learning performed in such a fashion has been shown, in most cases, to improve the utility of aggregate models over and beyond utilities obtainable from individual data sets.

Transforming the data set has a negligible cost. The main overhead comes from the larger-sized network that needs to be trained due to the additional layers (in ITNs and CNNs). Additionally, in collaborative learning, overhead stems from the sample size increasing with the number of data owners. Utility obtained from transformed data approaches that of non-private data training but there is still a gap. However, we have treated the machine learning algorithm as a black box; an avenue for worthwhile future research is to explore tailoring machine learning scripts to work better and more efficiently for transformed data. We note that our current implementation is un-optimized and there is significant room for improvement.

Another avenue of fruitful research is to explore computational security properties of transformation based learning. With a prior example from Fully Homomorphic Encryption, theoretically a transformation with zero utility loss but which is computationally-hard

to invert should exist. Our framework sheds light on some possible constructions that are worth exploring.

REFERENCES

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

[2] Eric Benhamou, Jamal Atif, and Rida Laraki. 2018. Operator norm upper bound for sub-Gaussian tailed random matrices. *Available at SSRN 3307071* (2018).

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

[4] Tiziano Bianchi, Valerio Bioglio, and Enrico Magli. 2015. Analysis of one-time random projections for privacy preserving compressed sensing. *IEEE Transactions on Information Forensics and Security* 11, 2 (2015), 313–327.

[5] Olivier Bousquet and André Elisseeff. 2002. Stability and generalization. *The Journal of Machine Learning Research* 2 (2002), 499–526.

[6] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. 2020. An Attack on InstaHide: Is Private Learning Possible with Instance Encoding? *arXiv preprint arXiv:2011.05315* (2020).

[7] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, 3 (2011).

[8] Sitan Chen, Zhao Song, and Danyang Zhuo. 2020. On InstaHide, Phase Retrieval, and Sparse Matrix Factorization. *arXiv preprint arXiv:2011.11181* (2020).

[9] Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 737–746.

[10] Wonwoo Cho and Nam Yul Yu. 2019. Secure and Efficient Compressed Sensing-Based Encryption With Sparse Matrices. *IEEE Transactions on Information Forensics and Security* 15 (2019), 1999–2011.

[11] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*. 1655–1658.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[15] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. 2009. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 381–390.

[16] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[17] Arnaud Fréville. 2004. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research* 155, 1 (2004), 1–21.

[18] Aritra Ghosh, Himanshu Kumar, and PS Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[19] O. Goldreich, S. Micali, and A. Wigderson. 1987. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*. 218–229.

[20] Thore Graepel, Kristin Lauter, and Michael Naehrig. 2012. ML confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*. Springer, 1–21.

[21] Moritz Hardt, Ben Recht, and Yoram Singer. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*. PMLR, 1225–1234.

[22] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. 2010. TASTY: tool for automating secure two-party computations. In *Proceedings of the 17th ACM conference on Computer and communications security*. 451–462.

[23] Daniel Hsu, Kevin Shi, and Xiaorui Sun. 2017. Linear regression without correspondence. *arXiv preprint arXiv:1705.07048* (2017).

[24] Baihe Huang, Zhao Song, Runzhou Tao, Ruizhe Zhang, and Danyang Zhuo. 2020. InstaHide's Sample Complexity When Mixing Two Private Images. *arXiv preprint arXiv:2011.11877* (2020).

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

[26] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1651–1669.
[27] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 619–631.
[28] Takahiro Maekawa, Takayuki Nakachi, Sayaka Shiota, and Hitoshi Kiya. 2018. Privacy-preserving SVM computing by using random unitary transformation. In *2018 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. IEEE, 146–150.
[29] Jiří Matoušek. 2008. On variants of the Johnson–Lindenstrauss lemma. *Random Structures & Algorithms* 33, 2 (2008), 142–156.
[30] Francesca Mignacco, Florent Krzakala, Yue Lu, Pierfrancesco Urbani, and Lenka Zdeborova. 2020. The role of regularization in classification of high-dimensional noisy Gaussian mixture. In *International Conference on Machine Learning*. PMLR, 6874–6883.
[31] Noman Mohammed, Rui Chen, Benjamin CM Fung, and Philip S Yu. 2011. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 493–501.
[32] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. 19–38. https://doi.org/10.1109/SP.2017.12
[33] Ibuki Nakamura, Yoshihide Tonomura, and Hitoshi Kiya. 2016. Unitary transform-based template protection and its application to l 2-norm minimization problems. *IEICE TRANSACTIONS on Information and Systems* 99, 1 (2016), 60–68.
[34] Ashwin Pananjady, Martin J Wainwright, and Thomas A Courtade. 2017. Linear regression with shuffled data: Statistical and computational limits of permutation recovery. *IEEE Transactions on Information Theory* 64, 5 (2017), 3286–3300.
[35] Ariel D Procaccia, Sashank J Reddi, and Nisarg Shah. 2012. A maximum likelihood approach for selecting sets of alternatives. *arXiv preprint arXiv:1210.4882* (2012).
[36] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. 2019. XONN: XNOR-based Oblivious Deep Neural Network Inference. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, Nadia Heninger and Patrick Traynor (Eds.). USENIX Association, 1501–1518.
[37] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.
[38] Mark Rudelson, Roman Vershynin, et al. 2013. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability* 18 (2013).
[39] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).
[40] Claude E Shannon. 1949. Communication theory of secrecy systems. *The Bell system technical journal* 28, 4 (1949), 656–715.
[41] Xiaoqiang Sun, Peng Zhang, Joseph K Liu, Jianping Yu, and Weixin Xie. 2018. Private machine learning classification based on fully homomorphic encryption. *IEEE Transactions on Emerging Topics in Computing* 8, 2 (2018), 352–364.
[42] Hassan Takabi, Ehsan Hesamifard, and Mehdi Ghasemi. 2016. Privacy preserving multi-party machine learning with homomorphic encryption. In *29th Annual Conference on Neural Information Processing Systems (NIPS)*.
[43] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. 2019. Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 9 (2019), 2917–2931.
[44] Jayakrishnan Unnikrishnan, Saeid Haghighatshoar, and Martin Vetterli. 2018. Unlabeled sensing with random linear measurements. *IEEE Transactions on Information Theory* 64, 5 (2018), 3237–3253.
[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[46] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*. PMLR, 6438–6447.
[47] Hanshen Xiao and Srinivas Devadas. 2021. DAUnTLeSS: Data Augmentation and Uniform Transformation for Learning with Scalability and Security. Cryptology ePrint Archive, Report 2021/201. (2021). https://eprint.iacr.org/2021/201.
[48] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.
[49] Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. Association for Computational Linguistics.
[50] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).
[51] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).
[52] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13001–13008.

A PROOF OF THEOREM 1

The proof of Theorem 1 is divided into two parts. First, to lower bound $\mathcal{I}(T^{-1}; Adv) = \mathcal{I}(W^{-1}, \sigma^{-1}; Adv)$, it is clear that

$$\begin{aligned}\mathcal{I}(W^{-1}, \sigma^{-1}; Adv) &= \mathcal{I}(\sigma^{-1}; Adv) + \mathcal{I}(W^{-1}; Adv|\sigma^{-1}) \\ &\geq \mathcal{I}(W^{-1}; Adv|\sigma^{-1}).\end{aligned} \tag{11}$$

Thus, it is sufficient to consider $\mathcal{I}(W^{-1}; Adv|\sigma^{-1})$. For any estimator Adv, we call it is successful if

$$\Pr_{x\sim P, W\sim Q}(\|Adv(xW) - x\| < \epsilon) \geq 1 - \delta. \tag{12}$$

Since W is uniformly selected, $\mathcal{I}(W^{-1}; Adv|\sigma^{-1}) = H(W^{-1}) - H(W^{-1}|Adv)$, $H(W^{-1}) = \log(3^{d^2} - S_c)$, where S_c is the number of matrices in $\{0, 1\}^{d\times d}$ where $\|W\| > c$. We apply the following lemma to upper bound S_c.

LEMMA A.1 ([2]). *For a square matrix $W \in \mathbb{R}^{d\times d}$, whose entry is i.i.d. selected from a sub-Gaussian of zero mean, there exists non-negative constants a and b, such that*

$$\Pr(\|W\| > \eta\sqrt{d}) \leq a \cdot e^{-\eta b d},$$

for any $\eta > a$.

With some calculation, when we select W randomly from $\{0, \pm 1\}^{d\times d}$,

$$\Pr(\|W\| > c\sqrt{d}) \leq e^{-log(3/2)cd}.$$

Thus, we have $S_c \leq 3^d \cdot e^{-log(3/2)cd}$. Now, to handle $H(W^{-1}|Adv)$, let us consider any two matrices W and W' where $E = W - W'$, and a predictor W_0^{-1} that Adv selects,

$$\Pr_{x\sim P}\left(\|(xW - xW')W_0^{-1}\| > 2\epsilon\right) > 2\delta,$$

then W_0^{-1} can successfully recover at most one of transformed data xW and xW'. Otherwise, if both $\Pr_{x\sim P}(\|xWW_0^{-1} - x\| < \epsilon) \geq 1 - \delta$ and $\Pr_{x\sim P}(\|xW'W_0^{-1} - x\| < \epsilon) \geq 1 - \delta$ hold, then

$$\Pr_{x\sim P}(\|x(W - W')W_0^{-1}\| < 2\epsilon) \geq 1 - 2\delta,$$

which gives a contradiction. Clearly, we know if the adversary precisely selects the $W_0^{-1} = W^{-1}$, then $xW \cdot W^{-1} = x$, where secret input can be perfectly recovered under weight matrix W. In the following, we set $W_0 = W^{-1}$, and see how many other transformations W' exist that cannot be attacked successfully by the adversary. We introduce the following Lemma.

LEMMA A.2 (HANSON–WRIGHT INEQUALITY [38]). *Let $x = (x_1, x_2, ..., x_d)$ be a random Gaussian vector with independent coordinates where $\mathbb{E}(x_i) = 0$ and $Var(x_i) = \tau^2$, then for a matrix $A \in \mathbb{R}^{d\times d}$ and any $t \geq 0$,*

$$\Pr(xAx^T - \mathbb{E}[xAx^T] < -t) \leq exp[-\frac{1}{2}\frac{t}{\tau^2\|A\|}]. \tag{13}$$

Here, $\|A\|$ is the l_2 norm of A.

Combining Lemma A.1 and A.2, we can lower bound the norm $\|x(W-W')W^{-1}\|$ by considering $\|xE\| \cdot \frac{1}{\|W\|}$. For the first part, $\|xE\|$, we have $\|E\| \leq \|W\| + \|W'\| \leq 2c\sqrt{d}$ for any W and W'. Now, we can lower bound the $\|xE\|$ with Lemma A.2. First, take $A = EE^T$ into (13), it is noted that $\mathbb{E}[xAx^T] = \mathbb{E}[\|xE\|^2] = \tau^2\|E\|_F^2$, where in our special case $\|E\|_F^2$ is at least the number of nonzero entries in E. With the bound on $\|E\|$, we have that

$$\Pr(\|xE\|^2 < \tau^2\|E\|_F^2 - t) \leq e^{-\frac{t}{8c^2d\tau^2}}. \tag{14}$$

On the other hand, for any given W, there are at most

$$\binom{d^2}{\beta d^2} \times 2^{\beta d^2}$$

many selections of W' such that $\|E\|_F^2 = \|W - W'\|_F^2 \leq \beta d^2$. Thus, we have at least $((1-e^{-log(3/2)cd}))3^{d^2} - \binom{d^2}{\beta d^2} \times 2^{\beta d^2})$ many selections of W' such that $\|E\|_F^2 > \beta d^2$. Now putting things together, if we set

$$\epsilon = \frac{\sqrt{\tau^2\beta d^2 - t}}{2c\sqrt{d}}, \delta \leq \frac{1 - e^{-\frac{t}{8c^2d\tau^2}}}{2},$$

then

$$\mathcal{I}(W^{-1}; Adv|\sigma^{-1}) \geq \log\left((1-e^{-log(3/2)cd})3^{d^2}\right) - \log\left(\binom{d^2}{\beta d^2} \times 2^{\beta d^2}\right).$$

The next part of the proof is to handle $\mathcal{I}(W^{-1}, \sigma^{-1}; \sigma(xW)) = \mathcal{I}(W^{-1}; \sigma(xW)) + \mathcal{I}(\sigma^{-1}; \sigma(xW)|W^{-1})$. In our assumption, σ is some deterministic function selected, and thus $\mathcal{I}(\sigma^{-1}; \sigma(xW)|W^{-1})$ equals 0. On the other hand,

$$\mathcal{I}(W^{-1}; \sigma(xW)) = \mathcal{H}(\sigma(xW)) - \mathcal{H}(\sigma(xW)|W) \leq \mathcal{H}(\sigma(xW)).$$

Thus, one can simply upper bound $\mathcal{H}(\sigma(xW))$ by $\sum_{i=1}^{d} \mathcal{H}(\sigma(xW)(i))$, where $\sigma(xW)(i)$ corresponds to the i-th coordinate of $\sigma(xW)$. If we ignore the negligible fraction of W where $\|W\| > c\sqrt{d}$, then each entry of W is i.i.d. selected and thus each coordinate of $\sigma(xW)$ is identically distributed as $\sigma(\langle x, v\rangle)$, where v is a random vector, each coordinate of which is i.i.d. selected from $\{0, \pm 1\}$. Thus, the distribution of $\langle x, v\rangle$ is equivalent to $\mathcal{N}(0, \tau\|v\|)$, where $\|v\|^2$ follows a Binomial distribution $B(d, 2/3)$, where $\Pr(\|v\|^2 = u) = \binom{u}{d}(2/3)^u(1/3)^{d-u}$. Therefore, let $Q_1(\cdot)$ be the probability density function (pdf) of $\mathcal{N}(0, \tau\|v\|)$ where $\|v\|^2 \sim B(d, 2/3)$ then

$$\mathcal{H}(\sigma(xW)) = -\sum_{o \in \sigma(\cdot)} p_1(\sigma^{-1}(o)) \log(p_1(\sigma^{-1}(o))), \tag{15}$$

where $p_1(\sigma^{-1}(o))$ corresponds to the probability of $Q_1(\cdot)$ on the support set of the inverse of $\sigma(\cdot)$ at point o.

B PROOF OF THEOREM 2

We rewrite the linear operators $(\tilde{W}_1, \tilde{W}_2, ..., \tilde{W}_p)$ on $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_p)$ as

$$\boldsymbol{x} \cdot W = \boldsymbol{x} \cdot \begin{bmatrix} \tilde{W}_1 & 0 & 0 \\ 0 & \tilde{W}_2 & 0 \\ ... & ... & ... \\ 0 & 0 & \tilde{W}_p \end{bmatrix}, \tag{16}$$

where compared to the weight matrix used in Theorem 1, now the weight matrix W for the Transformer network becomes a block diagonal matrix.

With a similar reasoning as the proof of Theorem 1, first

$$H(W) = pH(\tilde{W}_i) \geq p\log\left((1 - e^{-log(3/2)c\sqrt{p}d_p}) \cdot 3^{d_p^2}\right).$$

As for $H(W|Adv)$, it is noted that since $\|\tilde{W}_i\| \leq c\sqrt{d}$, the produced diagonal matrix also satisfies $\|W\| \leq c\sqrt{d}$. On the other hand, since there are only pd_p^2 entries we can select in W, for any given W, there are at most $\left(\binom{p \cdot d_p^2}{\beta d^2} \times 2^{\beta d^2}\right)$ many W' such that $\|W - W'\|_F^2 \leq \beta d^2$. Thus, if we set

$$\epsilon = \frac{\sqrt{\tau^2\beta d^2 - t}}{2c\sqrt{d}}, \delta \leq \frac{1 - e^{-\frac{t}{8c^2d\tau^2}}}{2},$$

then

$$\mathcal{I}(W^{-1}; Adv|\sigma^{-1}) \geq p\log\left((1-e^{-log(3/2)c\sqrt{p}d_p})3^{d_p^2}\right) - \log\left(\binom{p \cdot d_p^2}{\beta d^2} \times 2^{\beta d^2}\right).$$

As for $I(T^{-1}; T(x))$, it is clear that the transformations on each patch are independent and thus

$$\mathcal{I}(T^{-1}; \sigma(xW)) = \sum_{i=1}^{p} \mathcal{I}(T^{-1}; \sigma(\boldsymbol{x}_i\tilde{W}_i)).$$

On the other hand, as assumed, both $\tilde{W}_i$ and $\boldsymbol{x}_i$ are independent and identically distributed, respectively. Therefore, each $\mathcal{I}(T^{-1}; \sigma(\boldsymbol{x}_i\tilde{W}_i))$ equals

$$\mathcal{H}(\sigma(\boldsymbol{x}\tilde{W})) - \mathcal{H}(\sigma(\boldsymbol{x}\tilde{W})|\tilde{W}),$$

the case in Theorem 1 but replacing d by d_p. Thus, $I(T^{-1}; T(x))$ is still $O(d)$ for the transformation designed for transformer case.

C ANALYSIS OF PRIVATE TRANSFORM FOR CNN

Different from the transformation for Transformer network analyzed in Theorem 2, in a CNN, the patches can be overlapped. However, we can still rewrite the linear operator defined in (16), while W is not strictly block-wise diagonal and $W \in \mathbb{R}^{d \times k^2 p}$.

We give an example here. Imagine $\boldsymbol{x} = (x_1, x_2, x_3)$ where (x_1, x_2) forms the first patch and (x_2, x_3) forms the second one. Two independent matrices $\tilde{W}_1, \tilde{W}_2 \in \mathbb{R}^{2\times 2}$ are generated and we express the transformation as follows,

$$\boldsymbol{x} \cdot W = (x_1, x_2, x_3) \cdot \begin{bmatrix} w_{11} & w_{12} & 0 & 0 \\ w_{13} & w_{14} & w_{21} & w_{22} \\ 0 & 0 & w_{23} & w_{24} \end{bmatrix}. \tag{17}$$

once $\tilde{W}_1$ and $\tilde{W}_2$ are invertible, the right-hand inverse of W can be written as

$$W_R^{-1} = \begin{bmatrix} w_{11}^{-1} & w_{13}^{-1}/2 & 0 \\ w_{12}^{-1} & w_{14}^{-1}/2 & 0 \\ 0 & w_{21}^{-1}/2 & w_{23}^{-1} \\ 0 & w_{22}^{-1}/2 & w_{24}^{-1} \end{bmatrix}.$$

Here, w_{ij}^{-1} denotes the j-th entry of $\tilde{W}_i^{-1}$. It is easy to verify that $WW_R^{-1} = \boldsymbol{I}_d$. From the above example, as the number of overlapped patches increase, it corresponds to a larger weight W. Especially, if each entry will be included in at most r patches, the norm of W can be up to $c\sqrt{d}r$, where $\|\tilde{W}_i\| \leq c\sqrt{d}$. On the other hand, with more patches and a larger corresponding W, the freedom in generating W also increases accordingly, by a factor of about r compared to the W for non-overlapped patches shown in the proof of Theorem

2. This is because if each entry will appear in r patches, then the number of patches and the corresponding $\tilde{W}_i$ will increase by r times. Thus, with properly scaling the free parameter c, t and β, the security guarantee of transformations for overlapped patches is almost the same as Theorem 2 by replacing d_p with k^2, the size of kernel.

To be formal, in the CNN case described. Amongst those overlapped patches, each entry (pixel) of input x will be included in at most $(k/s)^2$ patches, and totally there are $p = (\frac{\sqrt{d}-k}{s}+1)^2$ patches. For each $\tilde{W}_i \in \mathbb{R}^{k^2\times k^2}$, if $\|\tilde{W}_i\| \le c\sqrt{d}$, then the composite matrix W satisfies $\|W\| \le (k/s)^2 c\sqrt{d}$.

Since there are $k^4 p$ many entries to be selected in W, we have

$$\mathcal{I}(T^{-1}; Adv|\sigma^{-1}) \ge p\log\left((1-e^{-log(3/2)c\sqrt{d}k})3^{k^4}\right) - \log\left(\binom{p\cdot k^4}{\beta d^2}\times 2^{\beta d^2}\right),$$

where

$$\epsilon = \frac{\sqrt{\tau^2\beta d^2 - t}}{2c(k/s)^2\sqrt{d}}, \delta \le \frac{1-e^{-\frac{t}{8c^2(k/s)^4 d\tau^2}}}{2}.$$

As for $I(T^{-1}; T(x))$, we cannot simply write it as the sum of $I(T^{-1}, \sigma(x_i\tilde{W}_i))$ since x_i are not independent due to overlapping. However, the upper bound

$$I(T^{-1}; T(x)) \le pk^2\mathcal{H}(\sigma(\langle\tilde{x}, v\rangle) + o(1),$$

still holds where $\tilde{x} \sim \mathcal{N}(0, \tau\boldsymbol{I}_{k^2})$ and $\|v\|^2$ follows a Binomial distribution $\mathcal{B}(k^2, 2/3)$. Therefore, $I(T^{-1}; T(x)) = O(pk^2)$ while $\mathcal{I}(T^{-1}; Adv|\sigma^{-1}) = \Theta(pk^4)$. When we scale the selection of (c, t, β) in Theorem 2 to be $c, (k/s)^4 t, (k/s)^4\beta$, it provides asymptotically the same bound as described in Theorem 2 after replacing the patch size d_p by k^2 for constant k/s.

D ADDITIONAL EXPERIMENTS AND EXPLANATION

Training Time: In the proposed Transformer networks and CNNs, we incorporate an additional fully-connected layer for each patch, and therefore the training time of each epoch accordingly increases as compared to that of the original network.

We take the regular Resnet 20 as an example. With 50,000 CIFAR-10 samples, each epoch takes 2.0 seconds. Now, consider the modified CNN architecture shown in Section 3.5 when $k = 4$ and $s = 4$. Each image is split into 64 patches and accordingly 64 fully-connected layers are added to process each patch, respectively. Under the same setup, in each epoch, the 64 fully-connected layers' training takes 7.5 seconds while the subsequent layers in the CNN take 2.3 seconds.

With further incorporation of Task Augmentation with 2 owners, compared to single-user private learning, the processing time almost doubles. Continuing with the above example, the number of additional fully-connected layers doubles to 128 and they take 12.9 seconds per epoch to train. Accordingly the subsequent CNN part takes 4.5 seconds per epoch to train.

Task Augmentation Implementation Details: During our experiment, we set the classification loss function to be the binary cross entropy on the expanded label, which is more numerically stable compared to a simple regression on 10×10 classes. The learning rate strategy in our collaborative CNN learning is set to be 0.1, 0.01 and 0.001 for 1-150, 151-250 and 251-300 epochs, respectively. We adopt the first Task Augmentation method described in Section 4 to implement Resnet20 and Resnet56, i.e., we aggregate two $32\times32\times3$ images into a form $64 \times 32 \times 3$.

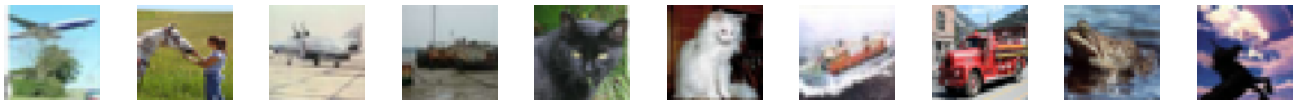
Multiple-layer Private Transformation and Recovery Attack: We provide another set of experiments where we consider more than one layer in the private transformation. The transformation design framework and PAC theory can be easily generalized to a multi-layer transformation associated with non-linear operators. For example, let the transformation $T(x) = \sigma(xW_1)\cdot W_2$, where W_i are still independent random $\{\pm1, 0\}^{d\times d}$ matrices and $\sigma(\cdot)$ is a normalized Sigmoid function $\sigma(z) = \frac{e^{z/10}}{1+e^{z/10}}$ in our following CIFAR-10 image recovery experiments.

Under the same setup, we assume 15,000 CIFAR transformed samples are exposed. Different prior knowledge of the adversary are assumed and shown in Fig. 7. The attack is implemented as described in Section 7.3, where a two-layer network formed by a fully-connected layer and a regression layer is trained to approximate the inversion. The two layers each have 3072 neurons and are connected by a Relu function.

The two-layer transformation $T(x) = \sigma(xW_1)\cdot W_2$ with a non-linear activation function σ imposes greater empirical hardness to invert the transformed pictures. On the other hand, the impact on training performance is within 1% compared to the single-layer transformation applied previously.

Figure 7: Additional Transformed CIFAR-10 Image Recovery

(a) Plain CIFAR-10 Images



(b) Recovery of FCN based Transformed Images with Prior Knowledge on 90% Correspondences and 4 Candidates Per Plain Sample

(c) Recovery of FCN based Transformed Images with Prior Knowledge on 50% Correspondences and 2 Candidates Per Plain Sample