Abstract

Cloud-based machine learning inference is an emerging paradigm where users query by sending their data through a service provider who runs an ML model on that data and returns back the answer. Due to increased concerns over data privacy, recent works have proposed Collaborative Inference (CI) to learn a privacy-preserving encoding of sensitive user data before it is shared with an untrusted service provider. Existing works so far evaluate the privacy of these encodings through empirical reconstruction attacks. In this work, we develop a new framework that provides formal privacy guarantees for an arbitrarily trained neural network by linking its local Lipschitz constant with its local sensitivity. To guarantee privacy using local sensitivity, we extend the Propose-Test-Release (PTR) framework to make it tractable for neural network queries. We verify the efficacy of our framework experimentally on real-world datasets and elucidate the role of Adversarial Representation Learning (ARL) in improving the privacy-utility trade-off.

1 Introduction

While training ML models privately has seen tremendous progress in the last few years, protecting privacy during the inference phase remains elusive as these models get deployed by cloud-based service providers. Cryptographic techniques address this challenge by computing over encrypted data. To combat the high computational cost of encryption techniques, several recent works have proposed Collaborative Inference (CI) – an alternate paradigm where users share a lower dimensional embedding of raw data where task-irrelevant information is suppressed. However, CI techniques are currently incompatible with formal privacy frameworks and their evaluation is currently based on empirical reconstruction attacks. The central issue in CI is the usage of Deep Neural Networks (DNNs) making it unsuitable for formal and worst-case privacy analysis. In this work, we take the first step towards a formal privacy framework for CI that will enable a more rigorous evaluation and a reliable understanding of the CI techniques.

The key aspect of any CI algorithm is an obfuscator (typically a DNN), which is trained to encode a user’s private data such that an attacker cannot reconstruct the original data from its encoding. Hence, CI techniques typically evaluate the privacy of their representations by empirically measuring the information leakage using a proxy adversary. However, existing works show that a proxy adversary’s performance as a measure of protection could be unreliable because a future adversary can come up with a stronger attack. Alternately a few existing CI techniques have used theoretical tools for measuring information leakage empirically. However, most of these works analyze specific obfuscation techniques and lack formal privacy definitions. Therefore, we first introduce a privacy definition applicable to CI by adopting a different threat model from that of differential privacy because, as we explain in detail in Sec 2, protecting membership inference is at odds with achieving a non-trivial privacy-utility trade-off in CI.
Our threat model for the reconstruction attack is motivated by the intuition that obfuscator discloses coarse-grained and filters out fine-grained information for preventing reconstruction. We hypothesize that to do so, the obfuscator should act as a contractive mapping, and as a result, increases the stability of the (obfuscation) function in the local neighborhood of data. We formalize this intuition by introducing a privacy definition in metric space of data and experimentally validate it in Sec[5]. Our privacy definition is an instantiation of metric-DP by Chatzikokolakis et al. Instead of evaluating the global Lipschitz constant of DNNs, we evaluate the Lipschitz constant only in the local neighborhood of the user’s sensitive data. We then extend the Propose-Test-Release (PTR)[11] framework to formalize our local neighborhood based measurement of the Lipschitz constant.

We adopt the PTR framework due to its posthoc and data-adaptive nature – as the privacy of encoding is evaluated in the neighborhood of the private user input and after the obfuscator has been trained. This approach is different from conventional privacy mechanisms where the mechanism is fixed and the perturbation does not depend on data. Existing seminal works in data-adaptive privacy such as Propose-Test-Release (PTR) [11], smooth sensitivity [30] and Lipschitz extensions [35] have exploited this posthoc approach to enhance the utility of the mechanisms. Data adaptive mechanisms are an important area of study as quoted in a recent workshop report on the challenges in Differential Privacy [9] - “Going data-adaptive is a crucial step in bringing DP algorithms to an acceptable level of utility in applications.”. In the context of our work – the benefit of using the PTR framework is two-folds i) Utility - the obfuscator is expected to behave better at samples that match the data distribution from its training dataset is sampled, and ii) Computation - instead of computing global sensitivity, we only need to compute sensitivity in the small neighborhood around private data points. While the original PTR is intractable due to a well-known [9] expensive computation step, we design a new formulation that substitutes the expensive computation step with a tractable alternate.

Computational tractability is generally the main concern that makes CI incompatible with formal privacy because giving worst-case guarantees for obfuscator is generally difficult. While training and convergence guarantees for DNNs remain elusive, we find that certified robustness community has made rapid progress in the past few years in giving worst-case guarantees during inference stage of DNNs. Specifically, we utilize recent efficient formulations for the Lipschitz constant computation of DNNs [19, 36]. We measure the stability of the learned obfuscator model using the Lipschitz constant and link it with the local sensitivity under our privacy definition. We bridge the gap between the Lipschitz constant and our extended PTR framework by proposing a binary search algorithm that computes this Lipschitz constant multiple times in the neighborhood of sensitive data. This bridge makes our work amenable to any differentiable obfuscator with ReLU non-linearity.

The scope of our paper is to provide privacy guarantees against reconstruction attacks for existing CI techniques, i.e., our goal is not to develop a new CI algorithm but rather to develop a formal privacy framework compatible with existing algorithms. A majority of the CI techniques protect either a sensitive attribute or reconstruction of the input. We only consider sensitive input in this work. Furthermore, we only focus on protecting the privacy of data during the inference stage, and assume that ML models can be trained privately. Our privacy definition and the mechanism are built upon $d_X$-privacy [8] and PTR [11]. Existing instantiations of $d_X$-privacy include geo-indistinguishability [2] and location-dependent privacy [24] that share a similar goal as ours of sharing coarse-grained information. Our work differs because we use DNN queries and high-dimensional datasets. We refer the reader to Supplementary for a detailed literature review.

In Sec[2] we begin with the preliminaries of DP and its variant for metric spaces. Then, we introduce the privacy definition relevant for CI in Sec[5]. Next, we design our framework by extending PTR and proving its privacy guarantees in Sec[4]. In Sec[5] we experimentally demonstrate the feasibility of our framework. We now give the summary of our contributions:

1. Proposing a privacy definition that formalizes reconstruction privacy in the context of CI. Protecting against membership inference (meaningfully) is not possible in CI and hence traditional DP can not be applied directly as discussed in Sec[2].
2. Extending the (PTR) algorithm to guarantee privacy using the local Lipschitz constant. The standard PTR can not be applied due to its intractability for queries beyond median and mode (see - Page 152, 2nd paragraph of Dwork et al. and Sec[4]).
3. Experiments 1) to verify efficacy for both utility and computability, 2) ablation on different design choices, 3) empirical reconstruction attacks, and 4) identifying the role ARL techniques play in improving privacy-utility trade-off in CI.
To formalize reconstruction privacy, we hypothesize that semantically similar points are close to each other on a data manifold, i.e. semantically similar samples are closer in a space where distance is defined as geodesic on the data manifold. Therefore, one way to bound the reconstruction of $x$ looking in inference. Intuitively, we choose a distance metric that provides indistinguishability only within a different distance metric yields a stronger or weaker privacy guarantee. Due to the impossibility that generalizes the DP definition to a general distance metric beyond hamming distance as follows:

$$
\Pr(M(x) \in S) \leq e^{d_H(x, x')} \Pr(M(x') \in S),
$$

where $d_H$ is the Hamming distance. This definition is based on a trusted central server model, where a trusted third party collects sensitive data and computes $M(x)$ to share with untrusted parties. In local-DP, this model has been extended such that each user shares $M(x)$, and the service provider is untrusted. Our threat model is the same as local DP. However, unlike local DP, data is not aggregated over multiple individuals. Specifically, every sample is evaluated independently by the service provider and there is no aggregation involved. For ex., a user shares a face image to receive an age prediction from the service provider, here the answer to the query depends exactly on the user’s input. With traditional local DP, it is “impossible” to achieve good utility and privacy at the same time because any two samples (no matter how different) are neighboring databases, more formally, $d_H(x, x') \leq 1$, $\forall x, x' \in \mathcal{X}$ for ML inference. Informally, this notion of neighboring databases under the standard DP definition would imply that the outcome for any two samples should be almost indistinguishable no matter how different the samples are. This privacy definition could be too restrictive for our ML inference application where the data instance necessarily needs a certain degree of distinguishability to obtain utility from the service provider. This observation is formalized in the impossibility result of instance encoding[7] for private learning. To subside this fundamental conflict between the privacy definition and our application, we look at the definition of $d_X$-privacy[8] that generalizes the DP definition to a general distance metric beyond Hamming distance as follows:

$$
\Pr(M(x) \in S) \leq e^{d_X(x, x')} \Pr(M(x') \in S),
$$

where $d_X(x, x')$ is a function that gives a level of indistinguishability between two datasets $x$ and $x'$.

DP can be viewed as a special case of $d_X$-privacy by keeping $d_X(x, x') = d_H(x, x')$. Choosing a different distance metric yields a stronger or weaker privacy guarantee. Due to the impossibility of achieving both privacy and utility with Hamming distance, we move towards a weaker privacy definition by only focusing on achieving privacy against reconstruction attacks instead of membership inference. Intuitively, we choose a distance metric that provides indistinguishability only within a neighborhood of similar looking samples.

3 Privacy Definition and Threat Model

To formalize reconstruction privacy, we hypothesize that semantically similar points are close to each other on a data manifold, i.e. semantically similar samples are closer in a space where distance is defined as geodesic on the data manifold. Therefore, one way to bound the reconstruction of $x$...
is by making it indistinguishable among semantically similar points. The extent of reconstruction privacy would therefore depend upon the radius of the neighborhood. We formalize it with a privacy parameter $R$ that allows a user to control how big this indistinguishable neighborhood should be.

**Definition 1.** A mechanism $\mathcal{M}$ satisfies $(\epsilon, \delta, R)$-reconstruction privacy if $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$ s.t. $d_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \leq R$ and $S \subseteq Range(\mathcal{M})$

$$P(\mathcal{M}(\mathbf{x}) \in S) \leq e^\epsilon P(\mathcal{M}(\mathbf{x}') \in S) + \delta.$$  \hfill (2)

Note that the above equation is exactly the same as $(\epsilon, \delta)$-DP except for the definition of neighboring databases. In this way, our privacy definition can be seen as a mix of standard DP and $d_{\mathcal{X}}$-privacy.

### 3.1 Threat and Attack model

Our threat and attack model is the same as previous collaborative inference works [44, 43]. Specifically, we focus on attackers that aim to reconstruct the original sample from its encoding. That is, given an encoding $\mathcal{M}(\mathbf{x})$ obtained from a sample $\mathbf{x}$, an attacker will attempt to recover the original sample $\mathbf{x}$. Our threat model is the same as local DP. Specifically, all side information including access to the obfuscator model, embedder model, and the framework in Fig 1 is known to the adversary and it can interact with them as much as it wants. The attacker can use this information to improve its attack algorithm. We (informally) note that an attacker always has a prior about raw data $P(\mathbf{x})$ and it will update its belief based on the posterior $P(\mathbf{x}|\mathcal{M}(\mathbf{x}))$. The goal of our privacy mechanism is to ensure that the distance between the prior and posterior distribution is bounded. Our definition indeed enjoys such an interpretation due to being an instantiation of $d_{\mathcal{X}}$-privacy. A more formal description of this Bayesian formulation is given by Chatzikokolakis et al. in Theorem 4. We emulate such attackers in Sec 5 by training an ML model on a dataset of $\mathcal{M}(\mathbf{x})$ as input and $\mathbf{x}$ as output, then we evaluate its reconstruction error on encodings of previously unseen $\mathcal{M}(\mathbf{x})$.

### 3.2 Comparison with Differential Privacy

Conceptually, the usage of hamming distance in DP for neighboring databases provides a level of protection at the sample level. Such a privacy requirement can be at odds with the goal of prediction that necessarily requires discrimination between samples belonging to different concept classes. We relax this dichotomy by changing the distance metric to account for reconstruction privacy. Intuitively, an attacker can only obtain an accurate reconstruction of data up to a neighborhood in the space of data. Therefore, the size of the neighborhood is a privacy parameter $R$ such that a higher value of $R$ provides higher privacy. This privacy parameter is equivalent to the group size [12] used often in the DP literature. By default, this value is kept at 1 in DP but can be kept higher if correlated individuals (multiple samples, family) are present in a database instead of a single individual. We state the equivalence between the group privacy definition and standard DP informally -

**Lemma 2.2 in [39]:** Any $(\epsilon, \delta)$-differentially private mechanism is $(R \epsilon, R e^{(R-1)\epsilon}/\delta)$-differentially private for groups of size $R$.

This lemma also applies to our proposed definition. However, we emphasize that privacy parameters of $(\epsilon, \delta)$-DP mechanism cannot be compared trivially with a $(\epsilon, \delta, R)$-reconstruction privacy mechanism because same value of $\epsilon$ and $\delta$ provide different levels of protection due to different definitions of neighboring databases. We experimentally demonstrate this claim in Sec.5.

### 3.3 Choice of $d_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')$ and its impact on the privacy guarantees offered

Our privacy definition is an instantiation of $d_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')$ that permits a broad class of distance metrics for the privacy definition. Since the goal of our work is to protect against reconstruction - $\ell_1$ or $\ell_2$ distance is a reasonable choice. However, the space over which $\ell$ norm should be considered is a challenging question. Existing papers have used metrics such as $\ell_1$ or $\ell_2$ norm in the raw data space [17, 6, 13, 3] for guaranteeing reconstruction privacy when training ML models. However, for high-dimensional datasets typically considered in CI such as images, guaranteeing reconstruction in ambient dimensions can lead to unintended privacy guarantee. In our work, we consider pre-processing the samples by projecting them into an embedding space, where samples that are semantically closer and farther have $\ell$ norm smaller and higher respectively. We evaluate the privacy-utility trade-off in our experiments for both ambient space and embedding space.
4 Privacy Mechanism

Our goal is to design a framework that can provide a formal privacy guarantee for encoding informally privatized in CI. However, CI algorithms use non-linear neural networks trained on non-convex objectives making it difficult to perform any worst-case analysis. Therefore, we take a posthoc approach where we reason about privacy after the model is trained. Specifically, we apply the PTR mechanism by [11]. Applying PTR directly to our query (ARL) is not computationally feasible because PTR requires estimating local sensitivity at multiple points whereas evaluating the local sensitivity of a neural network query is not even feasible at a single point. Therefore, we design a tractable variant of PTR that utilizes the local Lipschitz constant estimator to compute privacy related parameters. We refer the reader to supplementary for background on the Lipschitz constant estimation and PTR.

CI algorithms consist of three computational blocks in the training stage: 1) obfuscator \( f(\cdot) \) that generates an (informally private) representation \( z \) of data, 2) proxy adversary that reconstructs the data from the representation produced by the obfuscator, and 3) classifier that performs the given task using the obfuscated representation. The classifier and proxy adversary are trained to minimize the task loss and reconstruction loss, respectively. The obfuscator is trained to minimize the task loss but maximize the reconstruction loss. This setup results in a min-max optimization where the trade-off between task performance and reconstruction is controlled by a hyper-parameter \( \alpha \). Note that a few CI techniques[31, 33, 40] do not require a proxy adversary but still learn an obfuscator model using other regularizers.

At a high level, our framework applies the mechanism \( M \) such that the final released data \( \hat{x} = M(x) \) has a privacy guarantee discussed in Eq[2]. Like PTR, we start with a proposal \( (\Delta_{LS}^p) \) on the upper bound of the local sensitivity of x. To test the correctness of \( \Delta_{LS}^p \), we compute the size of the biggest possible neighborhood such that the local Lipschitz constant of the obfuscator network in that neighborhood is smaller than the proposed bound. Next, we privately verify the correctness of the proposed bound. We do not release the data (denoted by \( \perp \)) if the proposed bound is invalid. Otherwise, we perturb the data using the Laplace mechanism. Next, we discuss the framework and privacy guarantees in more detail.

Global Sensitivity and Lipschitz constant of a query \( f : X \rightarrow Y \) are related in the \( d_X \)-privacy framework. Global sensitivity of a query \( f(\cdot) \) is the smallest value of \( \Delta \) (if it exists) such that \( \forall x, x' \in X \), \( d_Y(f(x), f(x')) \leq \Delta d_X(x, x') \). While global sensitivity is a measure over all possible pairs of data in the data domain \( X \), local sensitivity \( (\Delta_{LS}) \) is defined with respect to a given dataset \( x \) such that \( \forall x' \in X \), \( d_Y(f(x), f(x')) \leq \Delta_{LS}(x)d_X(x, x') \). We integrate the notion of similarity in a neighborhood (motivated in Sec[2]) by defining the local sensitivity of a neighborhood \( N(x, R) \) around \( x \) of radius \( R \) such that \( \forall x' \in N(x, R) \) the local sensitivity of query \( f \) on \( x \) in the neighborhood is defined \( \forall x' \in N(x, R) \) such that

\[
d_Y(f(x), f(x')) \leq \Delta_{LS}(x, R)d_X(x, x').
\] (3)

We note that if \( d_X \) is hamming distance and \( R \) is 1 then this formulation is exactly same as local sensitivity in \( \epsilon \)-DP[12]. The equation above can be re-written as:

\[
\Delta_{LS}(x, R) = \sup_{x' \in N(x, R)} \frac{d_Y(f(x), f(x'))}{d_X(x, x')}.
\] (4)

This formulation of local sensitivity is similar to the definition of the local Lipschitz constant. The local Lipschitz constant \( \mathcal{L} \) of \( f \) for a given open neighborhood \( N \subseteq X \) is defined as follows:

\[
\mathcal{L}^{\alpha, \beta}(f, N) = \sup_{x', x'' \in N} \frac{\| f(x') - f(x'') \|_\alpha}{\| x' - x'' \|_\beta} \quad (x' \neq x'')
\] (5)

We note that while the local sensitivity of \( x \) is described around the neighborhood of \( x \), the Lipschitz constant is defined for every possible pair of points in a given neighborhood. Therefore, in Lemma[4.1] we show that the local Lipschitz in the neighborhood of \( x \) is an upper bound on the local sensitivity.

**Lemma 4.1.** For a given \( \ell \) and for \( d_Y \leftarrow \ell_\alpha \) and \( d_X \leftarrow \ell_\beta \), \( \Delta_{LS}(x) \leq \mathcal{L}(f, N(x, R)) \). Proof in Supplementary.

Since local sensitivity is upper bounded by the Lipschitz constant, evaluating the Lipschitz constant suffices as an alternative to evaluating local sensitivity.
Lower bound on testing the validity of $\Delta_{LS}^p$: The PTR algorithm[11] suggests a proposal on the upper bound ($\Delta_{LS}^p$) of local sensitivity and then finds the distance between the given dataset ($x$) and the closest dataset for which the proposed upper bound is not valid. Let $\gamma(\cdot)$ be a distance query and $\Delta_{LS}(x)$ be the local sensitivity defined as per the DP framework with respect to $x$ such that

$$\gamma(x) = \min_{x' \in \mathcal{X}} \{d_H(x, x') \ s.t. \ \Delta_{LS}(x') > \Delta_{LS}^p\}.$$ (6)

For our privacy definition, the query $\gamma(x, R)$ can be formulated as follows:

$$\gamma(x, R) = \min_{x' \in \mathcal{X}} \{d_H(x, x') \ s.t. \ \Delta_{LS}(x', R) > \Delta_{LS}^p\}.$$ (7)

We note that keeping $d_x = d_H$ and $R = 1$, makes the query $\gamma(x)$ exactly same as defined in the equation above. Generally, computing $\gamma(\cdot)$ is intractable due to local sensitivity estimation required for every $x' \in \mathcal{X}$ (which depends upon a non-linear neural network in our case). We emphasize that this step is intractable at two levels, first, we require estimating the local sensitivity of a neural network query. Second, we require this local sensitivity over all samples in the data domain. Therefore, we make it tractable by computing a lower bound over $\gamma(x, R)$ by designing a function $\phi(\cdot)$ s.t.

$$\phi(x, R) \leq \gamma(x, R).$$

Intuitively, $\phi(\cdot)$ finds the largest possible neighborhood around $x$ such that the local Lipschitz constant of the neighborhood is smaller than the proposed local sensitivity. Because the subset of points around $x$ whose neighborhood does not violate $\Delta_{LS}^p$ is half of the size of the original neighborhood in the worst case, we return half of the size of neighborhood as the output. We describe the algorithm procedurally in the supplementary material. More formally,

$$\phi(x, R) = \frac{1}{2} \cdot \arg \max_{R' \geq R} \{L(f, N'(x, R')) \leq \Delta_{LS}^p\}.$$

If there is no solution to the equation above, then we return 0.

**Lemma 4.2.** $\phi(x, R) \leq \gamma(x, R)$. Proof in Supplementary.

Privately testing the lower bound: The next step in the PTR algorithm requires testing if $\gamma(x) \leq \ln\left(\frac{1}{\epsilon}\right)$. If the condition is true, then no-answer ($\perp$) is released instead of data. Since the query depends upon $x$, PTR privatizes it by applying laplace mechanism, i.e. $\hat{\gamma}(x) = \gamma(x) + \text{Lap}(1/\epsilon)$. The query has a sensitivity of 1 since the $\gamma$ could differ at most by 1 for any two neighboring databases.

In our framework, we compute $\phi(x, R)$ to lower bound the value of $\gamma(x, R)$. Therefore, we need to privatize the $\phi$ query. We prove that for distance metrics in $d_x$-privacy, the global sensitivity of the $\phi(x)$ query is 1.

**Lemma 4.3.** The query $\phi(\cdot)$ has a global sensitivity of 1, i.e. $\forall x, x' \in \mathcal{X}$, $d_{\text{abs}}(\phi(x, R), \phi(x', R)) \leq d_x(x, x')$. Proof in Supplementary.

After computing $\phi(x, R)$, we add noise sampled from a laplace distribution, i.e. $\hat{\phi}(x, R) = \phi(x, R) + \text{Lap}(R/\epsilon)$. Next, we check if $\hat{\phi}(x, R) \leq \ln\left(\frac{1}{\epsilon}\right) \cdot R/\epsilon$, then we release $\perp$, otherwise we release $\hat{z} = f(g(x)) + \text{Lap}(\Delta_{LS}^p/\epsilon)$. Next, we prove that the mechanism $\mathcal{M}_1$ described above satisfies reconstruction-privacy.

**Theorem 4.4.** Mechanism $\mathcal{M}_1$ satisfies uniform $(2\epsilon, \delta/2, R)$-reconstruction privacy Eq. 2 i.e.

$$\forall x, x' \in \mathcal{X}, s.t. \ d_x(x, x') \leq R$$

$$\mathbb{P}(\mathcal{M}(x) \in S) \leq e^{2\epsilon} \mathbb{P}(\mathcal{M}(x') \in S) + \frac{\delta}{2}$$ (8)

Proof Sketch: Our proof is similar to the proof for the PTR framework[12] except the peculiarity introduced due to our metric space formulation. First, we show that not releasing the answer ($\perp$) satisfies the privacy definition. Next, we divide the proof into two parts, when the proposed bound is incorrect (i.e. $\Delta_{LS}(x, R) > \Delta_{LS}^p$) and when it is correct. Let $\hat{R}$ be the output of query $\phi$.

$$\mathbb{P}[\hat{\phi}(x, R) = \hat{R}] = \exp\left(-\frac{\|\phi(x, R) - \hat{R}\|}{R} \cdot \epsilon\right)$$
$$\mathbb{P}[\phi(x', R) = \hat{R}] = \exp\left(-\frac{\|\phi(x', R) - \hat{R}\|}{R} \cdot \epsilon\right)$$

$$\leq \exp(\|\phi(x', R) - \phi(x, R)\| \cdot \frac{\epsilon}{R}) \leq \exp(d_x(x, x') \cdot \frac{\epsilon}{R}) \leq \exp(\epsilon)$$
Table 1: **Performance comparison for different baselines**: Our posthoc framework enables comparison between different obfuscation techniques by fixing the privacy budget (\(\epsilon\)). First four rows are different approaches to protect against data reconstruction and the remaining rows below are combinations of different approaches. The top row refers to the accuracy corresponding to different datasets under two extremes of epsilons. ARL refers to widely used adversarial representation learning approach for regularizing representation based on a proxy attacker [26,27,53,37]. Contrastive refers to contrastive learning based informally privatizing mechanism introduced in [53].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Encoder</th>
<th>ARL-C</th>
<th>ARL-N</th>
<th>C-N</th>
<th>ARL-C-N</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST ((\epsilon = 0.10), (\epsilon = \infty), 0.93)</td>
<td>0.93</td>
<td>0.917</td>
<td>0.896</td>
<td>0.92</td>
<td>0.921</td>
</tr>
<tr>
<td>FMNIST ((\epsilon = 0.10), (\epsilon = \infty), 0.781)</td>
<td>0.79</td>
<td>0.747</td>
<td>0.761</td>
<td>0.77</td>
<td>0.706</td>
</tr>
<tr>
<td>UTKFace ((\epsilon = 0.502), (\epsilon = \infty), 0.732)</td>
<td>0.77</td>
<td>0.712</td>
<td>0.71</td>
<td>0.712</td>
<td>0.650</td>
</tr>
</tbody>
</table>

Therefore, using the post-processing property - \(\mathbb{P}[\mathcal{M}(\mathbf{x}) = \perp] \leq e^{\epsilon}\mathbb{P}[\mathcal{M}(\mathbf{x}') = \perp] \). Here, the first inequality is due to triangle inequality, the second one is due to Lemma 4.3 and the third inequality follows from \(d_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \leq R\). Note that when \(\Delta_{L,S}(\mathbf{x}, R) > \Delta_{LS}^R\), \(\phi(\mathbf{x}, R) = 0\). Therefore, the probability for the test to release the answer in this case is

\[
\mathbb{P}[\mathcal{M}(\mathbf{x}) \neq \perp] = \mathbb{P}[\phi(\mathbf{x}, R) + \text{Lap}(\frac{R}{\epsilon}) > \log(\frac{1}{\delta}) \cdot R] = \mathbb{P}[\text{Lap}(\frac{R}{\epsilon}) > \log(\frac{1}{\delta}) \cdot R]
\]

Based on the CDF of Laplace distribution, \(\mathbb{P}[\mathcal{M}(\mathbf{x}) \neq \perp] = \frac{\delta}{2}\). Therefore, if \(\Delta_{L,S}(\mathbf{x}, R) > \Delta_{LS}^R\), for any \(S \subseteq \mathbb{R}^d \cup \perp\) in the output space of \(\mathcal{M}\)

\[
\mathbb{P}[\mathcal{M}(\mathbf{x}) \in S] = \mathbb{P}[\mathcal{M}(\mathbf{x}) \in S \cap \{\perp\}] + \mathbb{P}[\mathcal{M}(\mathbf{x}) \in S \cap \{\mathbb{R}^d\}] \\
\leq e^{\epsilon}\mathbb{P}[\mathcal{M}(\mathbf{x}') \in S \cap \{\perp\}] + \mathbb{P}[\mathcal{M}(\mathbf{x}) \neq \perp] \leq e^{\epsilon}\mathbb{P}[\mathcal{M}(\mathbf{x}') \in S] + \frac{\delta}{2}
\]

If \(\Delta_{L,S}(\mathbf{x}, R) \leq \Delta_{LS}^R\) then the mechanism is a composition of two \((\epsilon, \delta, R)\)-reconstruction private algorithm where the first algorithm \((\phi(\mathbf{x}, R))\) is \((\epsilon, \delta/2, R)\)-reconstruction private and the second algorithm is \((\epsilon, 0, R)\)-reconstruction private. Using composition, the algorithm is \((2\epsilon, \delta/2, R)\)-reconstruction private. We describe \(M_1\) step by step in the algorithm in supplementary. To summarize, we designed the posthoc privacy framework that extends the PTR framework by making it tractable to get \((\epsilon, \delta, R)\)-reconstruction privacy. The exact local Lipschitz constant of the neural network based obfuscator is estimated using mixed-integer programming based optimization developed by [19].

**Computational feasibility**: Our key idea is to add extra computation on the client side to turn informally private representations to their formal counterpart. This extra computational cost is due to the estimation of the local Lipschitz constant of the obfuscator network. However, the following two factors of our framework make it practically feasible -

1. We compute the local Lipschitz constant (i.e. in a small neighborhood around a given point):
   Our extension of the propose-test-release framework only requires us to operate in a small local neighborhood instead of estimating the global Lipschitz constant which would be much more computationally expensive.

2. Low number of parameters for obfuscator: Instead of estimating the Lipschitz constant of the whole prediction model, we only require estimation of the obfuscator - a neural network that has a significantly lower number of parameters in comparison to the prediction model.

We experimentally validate both of the above benefits in Sec 5. The fact that the local Lipschitz constant is being computed over the same obfuscator allows room for optimizing performance by caching. Our goal is to demonstrate the feasibility of bridging formal privacy guarantees and CI mechanisms, hence, we did not explore such performance speedups.
5 Experimental Setup: We evaluate different aspects of our proposed framework - i) E1: comparison between different adversarial approaches, ii) E2: comparison with local differential privacy (LDP), iii) E3: computational tractability of our proposed framework, and iv) E4: investigating role of ARL in achieving privacy. We use MNIST [25], FMNIST [42] and UTKFace [45] dataset for all experiments. All of them contain samples with high ambient dimensions (MNIST-784, FMNIST-784 and UTKFace-4096). We use a deep CNN based β-VAE [18] for the pre-processing. We use LipMip [19] for computing Lipschitz constant over $\ell_\infty$ norm in the input space and $\ell_1$ norm in the output space. Our baselines include a simple Encoder that projects the data to smaller dimensions using a neural network. This encoder type approach has been used in the literature as Split Learning [15]. We include widely used ARL techniques [43, 37, 27, 26, 44] and adversarial contrastive learning [33] which we denote as C. We use noisy regularization (denoted by N) to improve classifier performance. We refer the reader to supplementary for a detailed experimental setup, codebase, and hyper-parameters.

E1: Privacy-Utility Trade-off: Since our framework enables comparison between different obfuscation techniques under same privacy budget, we evaluate test set accuracy as utility in Table 1. Our results indicate that ARL complemented with contrastive and noise regularization helps in attaining overall best performance among all possible combinations.

E2: Comparison between ARL and LDP: While $\epsilon$-LDP definition provides a different privacy guarantee than our proposed privacy definition, for reference we compare the performance between LDP and CI and report results in supplementary. We note that for low values of $\epsilon$, LDP techniques do not yield any practical utility. This corroborates with the impossibility result of instance encoding [7] and our discussion in Sec 2 about the inapplicability of traditional DP in the context of CI.

E3: Computational feasibility: We report end-to-end runtime evaluation on a CPU-based client and achieve 2 sec/image (MNIST) and 3.5 sec/image (UTKFace). While plenty of room exists for optimizing this runtime, we believe current numbers serve as a reasonable starting point for providing formal privacy in ARL. As discussed in Sec 1 we compare the computation time of the obfuscator across three factors relevant to our setup - i) Dimensionality of the input, ii) Size of the neighborhood, and iii) Number of layers in the Obfuscator. We report the results in Fig. 2.

E4: What role does ARL play in achieving privacy? We investigate the contribution of adversarial training in improving the privacy-utility trade-off. We train obfuscator models with different values of $\alpha$ (weighing factor) for adversarial training. Our results in Fig 3 indicate that higher weighing of adversarial regularization reduces the local Lipschitz constant, hence reducing the local sensitivity of the neural network. Furthermore, for high values of $\alpha$, the change in the local Lipschitz constant reduces significantly for different sizes ($R$) of the neighborhood. These two observations can potentially explain that ARL improves reconstruction privacy by reducing the sensitivity of the obfuscator. However, as we observe in Table 1, the classifier can reduce its utility if ARL is not complemented with noisy and contrastive regularization. We believe this finding could be of independent interest to the adversarial defense community.
6 Discussion

**How to select privacy parameter R?** One of the key differences between ($\epsilon$, $\delta$, $R$)-neighborhood privacy and ($\epsilon$, $\delta$)-DP is the additional parameter $R$. The choice of $R$ depends upon the neighborhood in which a user wishes to get an $\epsilon$ level of indistinguishability. We discussed the equivalence of $R$ and $k$ in group Differential Privacy in Sec 3.2. To understand the role of $R$, we perform reconstruction attacks on privatized encoding obtained from our framework by training an ML model to reconstruct original images. We compare reconstruction results for different values of $\epsilon$ and $R$ on four distinct metrics for images and report in Supplementary. To assess the level of indistinguishability, we project the original images into embedding space and sample points from the boundary of neighborhoods of different $R$. We observe that as the boundary of the neighborhood increases, the images become perceptually different from the original image. For extremely large radii, the images change significantly enough that their corresponding label may change too. Such visualizations can be used to semantically understand different values of $R$.

**How to propose $\Delta_p^{LS}$?** Our framework requires a proposal on the upper bound of local sensitivity in a private manner. One possible way to obtain $\Delta^{LS}_p$ is by using the Lipschitz constant of training data samples used in training the obfuscator. We choose $\Delta^{LS}_p$ by computing the mean ($\mu$) and standard deviation ($\sigma$) of local sensitivity on the training dataset (assumed to be known under our threat model), then we keep $\Delta^{LS}_p = \mu + n \cdot \sigma$ where $n$ allows a trade-off between the likelihood of releasing the samples under PTR and adding extra noise to data. We used $n = 3$ in our experiments. Since empirically, the value of local sensitivity appears to be following a gaussian, using confidence interval serves as a good proxy. Fig 3 shows that for higher values of $\alpha$, the variability in the local Lipschitz constant decreases indicating the validity of the bound would hold for a large number of samples.

**Limitations:** i) Since we utilize the PTR framework, outlier samples may not get released due to high sensitivity, this is expected since these outlier samples are likely to be misclassified anyway. ii) Lipschitz constant computation is limited to ReLU based DNNs, therefore more sophisticated obfuscator architectures are currently not compatible with our proposed framework and we leave it as part of the future work. iii) Choosing the privacy parameter ($R$) could be challenging for certain datasets and might vary based on user preferences. We believe the choice of such a parameter would depend upon the context in which CI is being applied.

7 Conclusion

In this work we take a first step towards bridging empirical techniques in ARL and formal privacy mechanisms. The posthoc nature of our framework allows reasoning about privacy without any modification to the obfuscation algorithm, hence, making it easy to integrate for benchmarking existing and future techniques. We introduced a privacy definition that formalizes ARL and designed a corresponding privacy mechanism. An exciting future direction for extending our framework is to integrate NNs for designing more effective privacy mechanisms for other sophisticated queries beyond CI. Data adaptive mechanisms are an exciting area in DP and our improvement to PTR can be potentially applied for other queries where PTR might be intractable currently.
References


Supplementary Material: Posthoc privacy guarantees for collaborative inference with modified Propose-Test-Release

Anonymous Author(s)

Affiliation

Address

email

A Related Work

Collaborative Inference techniques aim to learn a task-oriented privacy preserving encoding of data. Majority of the works in this area either protect against sensitive attribute leakage [10, 28, 1, 19] or input reconstruction [30, 33, 22, 18, 21]. These techniques usually evaluate their privacy using empirical attacks since the mechanism is learned using gradient based min-max optimization making it infeasible for the worst-case privacy analysis. The goal of our work is to make them amenable to formal privacy analysis. While theoretical analyses [37, 38, 29] of ARL objectives have identified fundamental trade-offs between utility and attribute leakage, they are difficult to formalize as a worst-case privacy guarantee especially for deep neural networks.

Privacy definitions that extend the DP definition to incorporate some of its limitations [14] include $d_X$-Privacy [3], and Pufferfish [15]. Our privacy definition is a specific instantiation of the $d_X$-Privacy [3] framework that extends DP to general metric spaces. Our instantiation is focused on reconstruction privacy for individual samples instead of membership inference attacks [5]. Existing works in DP for reconstruction attacks [2, 34] focus on the privacy of training data.

Lipschitz constant estimation for neural networks has been used to guarantee network’s stability to perturbations. Existing works either provide an upper bound [36, 17, 6], exact Lipschitz constant [13, 12] or Lipschitz constant regularization [31, 11] during the training stage. Some existing works have explored the relationship between adversarial robustness and DP model training [25, 26, 35]. We utilize similar ideas of perturbation stability but for privacy. Shavit and Gjura [32] use Lipschitz neural networks [8] to learn a private mechanism design for summary statistics such as median, however their mechanism design lack privacy guarantee.

Posthoc approach to privacy applies privacy preserving mechanism in a data dependent manner. Smooth sensitivity [23] and PTR [4] reduce the noise magnitude since local sensitivity is only equal to global sensitivity in the worst case. Privacy odometer [27], Ex-post privacy loss [20] and Rényi privacy filter [7] track privacy loss as the query is applied over data. Our works builds upon the PTR framework in order to give high privacy for less sensitive data. However, as we show in Sec ??, our framework reformulates the PTR algorithm to make it tractable under our setup.

B Proofs

Lemma B.1. For a given $f$ and for $d_Y \leftarrow \ell_\alpha$ and $d_X \leftarrow \ell_\beta$, $\Delta_{LS}(x, R) \leq \mathcal{L}(f, \mathcal{N}(x, R))$.

Proof. Local sensitivity ($\Delta_{LS}$) for a sample $x$ in a radius $R$ for a query $f$ is defined as:

$$\Delta_{LS}(x, R) = \sup_{x' \in \mathcal{N}(x, R)} \frac{d_Y(f(x), f(x'))}{d_X(x, x')}$$

Local Lipschitz constant \( (L) \) for a function \( f \) and a neighborhood \( N \) is defined as:
\[
\mathcal{L}^{\alpha, \beta}(f, N) = \sup_{x', x'' \in N} \frac{||f(x') - f(x'')||_\alpha}{||x' - x''||_\beta} \quad (x' \neq x'')
\]

If \( L \) is defined around neighborhood \( N(x, R) \) then the set over which local sensitivity is computed is a subset of the set over which local Lipschitz constant is estimated. Intuitively, local Lipschitz condition is for all possible pair of samples in the neighborhood while local sensitivity is for all samples with respect to the given sample. Since both conditions require a supremum over the set, \( \Delta_{LS}(x, R) \leq L(f, N(x, R)) \).

**Lemma B.2.** Algorithm \( \phi \) gives a lower bound on the query \( \gamma \). That is, \( \phi(x, R) \leq \gamma(x, R) \).

**Proof.** The \( \gamma \) query is defined as -
\[
\gamma(x, R) = \min_{x' \in \mathcal{X}} \{d_X(x, x') \mid \Delta_{LS}(x', R) > \Delta_{LS}^{p}\}
\] (1)

The \( \phi \) query is defined as -
\[
\phi(x, R) = \frac{1}{2} \cdot \arg\max_{R' \geq R} \{\mathcal{L}(f, N(x, R')) \leq \Delta_{LS}^{p}\}
\] (2)

For any given sample \( x \) and privacy parameters \( (R, \Delta_{LS}^{p}) \) such that \( s = \phi(x, R) \), we know that \( \forall x' \in N(x, s) \)
\[
N(x', s) \subseteq N(x, 2s)
\]
\[
\implies \mathcal{L}(f, N(x', s)) \leq \mathcal{L}(f, N(x, 2s))
\]

Based on eq[1] we know that \( \mathcal{L}(f, N(x, 2s)) \leq \Delta_{LS}^{p} \) and hence \( \forall x' \in N(x, s) \),
\[
\mathcal{L}(f, N(x', s)) \leq \Delta_{LS}^{p}
\]

Therefore, \( \Delta_{LS}(x', R) \leq \Delta_{LS}^{p} \) and hence,
\[
s \leq \gamma(x)
\]

For the cases when there is not any feasible solution, \( \phi \) returns 0 which is exactly the same answer for \( \gamma \) query. This completes the proof.

**Lemma B.3.** The query \( \phi(\cdot) \) has a global sensitivity of 1, i.e. \( d_{abs}(\phi(x, R), \phi(x', R)) \leq d_X(x, x') \)

**Proof.** We will prove the above argument through a contradiction. We will prove that for a fixed radius \( R \) and any arbitrary point \( x \in \mathcal{X} \), the neighborhood spanned by \( \phi(x, R) \) can not be a proper superset for any neighborhood spanned by any other point \( \phi(x', R) \). More formally, we will prove, \( \forall x, x' \in \mathcal{X}, \overline{N(x, \phi(x, R))} \not\subseteq N(x', \phi(x', R)) \) and \( \overline{N(x', \phi(x', R))} \not\subseteq N(x, \phi(x, R)) \). Once proven, this argument allows us to specify the distance between \( x \) and \( x' \) with respect to \( \phi(x, R) \) and \( \phi(x', R) \).

Since the function \( \phi(x, R) \) returns the maximum possible value such that
\[
\mathcal{L}(f, N(X, \phi(x, R))) \leq \Delta_{LS}^{p}
\]

Therefore, for any \( \zeta > 0 \)
\[
\mathcal{L}(f, N(X, \phi(x, R) + \zeta)) > \Delta_{LS}^{p}
\]
(3)

For contradiction, we assume that \( \exists x, x' \in \mathcal{X} \) such that \( N(x, \phi(x, R)) \subseteq N(x', \phi(x', R)) \)
\[
\implies \exists \eta > 0 \text{ s.t. } N(x, \phi(x, R) + \eta) \subseteq N(x', \phi(x', R))
\] (4)
\[
\implies \mathcal{L}(N(x, \phi(x, R) + \eta)) \leq \Delta_{LS}^{p}
\] (5)

This leads to a contradiction between eq[3] and eq[5]. Therefore, \( \forall x, x' \in \mathcal{X} \),
\[
\phi(x, R) \leq \phi(x', R) + d_X(x, x')
\]

Using symmetry argument, we can show that
\[
d_{abs}(\phi(x, R), \phi(x', R)) \leq d_X(x, x')
\]

This completes the proof.
Algorithm 1: Extended PTR algorithm for \((\epsilon, \delta, R)\)-semantic neighborhood privacy

**Input:** \(x \in \mathcal{X}, \Delta_{LS}^p \in \mathbb{R}^+\);
**Params:** \(\epsilon \in \mathbb{R}^+, \delta \in \mathbb{R}^+, R \in \mathbb{R}^+\);
**Init:** \(\zeta \in \mathbb{R}^+\); // For numerical stability, typically very small
**Init:** \(R_{min} = R, R_{max} \in \mathbb{R}^+\);

repeat
\(R_{mid} = (R_{min} + R_{max})/2\);
\(r = L(f; \mathcal{N}(x, R))\); // Compute local Lipschitz constant
if \(r < \Delta_{LS}^p\) then
\(R_{min} = R_{mid}\);
else
\(R_{max} = R_{mid}\);
end if
until \(R_{max} > R_{min} + \zeta\)
\(\hat{r} = \frac{R_{min}}{2}\);
\(\hat{R} \leftarrow \hat{r} + \text{Lap}(1/\epsilon)\);
if \(\hat{R} < \ln(1/\delta)/\epsilon\) then
Return \(\bot\);
else
Return \(f(x) + \text{Lap}(\Delta_{LS}^p/\epsilon)\);
end if

C Propose-Test-Release

DP mechanisms typically add noise based on the global sensitivity of a query. However, for several queries over various data distributions, average local sensitivity might be much lower than the global sensitivity. However, local sensitivity is data dependent hence the amount of noise introduced by a mechanism based on local sensitivity itself can reveal private information. Therefore, to add noise based on local sensitivity in a privacy preserving manner, [4] introduced PTR. Conceptually, the idea behind PTR is to propose an arbitrary upper bound on the true value of local sensitivity. This upper bound should be obtained privately otherwise the choice of upper bound itself can reveal private information. To test whether the proposed bound is correct, the mechanism performs a privacy-preserving testing of the upper bound. The test itself is a randomized algorithm due to privacy requirements. Therefore, it can have false positives and false negatives. If the test fails, the mechanism returns \(\bot\) (no-answer). Otherwise, standard DP mechanism (ex. - Laplace) is applied to the query based on the proposed sensitivity (and not true local sensitivity). More formally, the algorithm proceeds in the following steps -

1. A proposal on upper bound of a query \(q\) is fed as input for data \(x\). Let us call it \(\Delta_{LS}^p\).
2. The algorithm finds the closest point \(x'\) to \(x\) such that \(\Delta_{LS}(x') > \Delta_{LS}^p\). Here \(\Delta_{LS}\) refers to local sensitivity for the query \(q\).
3. Let \(\gamma = d_H(x, x')\) and \(\hat{\gamma} = \gamma + \text{Lap}(1/\epsilon)\)
4. If \(\hat{\gamma} \leq \ln(1/\delta)/\epsilon\) then return \(\bot\)
5. Else share data \(x + \text{Lap}(\Delta_{LS}^p/\epsilon)\)

**Computational Cost:** Depending on the query, this algorithm can incur significant computation cost. Especially in the Step 2, finding closest \(x'\) can be impractical if the data space is high dimensional. Finally, for queries such as neural networks evaluating local sensitivity itself is not practical since it requires giving exact and correct solution to a non-convex optimization problem. Therefore, our framework relies on computing local lipschitz constant over a small neighborhood instead of local sensitivity over the complete data space.
$$\epsilon = 1 \quad \epsilon = 2 \quad \epsilon = 5 \quad \epsilon = 7 \quad \epsilon = \infty$$

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LDP-Image</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>LDP Embedding</td>
<td>0.1075</td>
<td>0.1219</td>
<td>0.1927</td>
<td>0.3107</td>
</tr>
<tr>
<td></td>
<td>Adversarial</td>
<td>0.514</td>
<td>0.751</td>
<td>0.891</td>
<td>0.912</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>F MNIST</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LDP-Image</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>LDP Embedding</td>
<td>0.1012</td>
<td>0.1251</td>
<td>0.2420</td>
<td>0.7798</td>
</tr>
<tr>
<td></td>
<td>Adversarial</td>
<td>0.514</td>
<td>0.751</td>
<td>0.891</td>
<td>0.912</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>UTKFace</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LDP-Image</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>LDP Embedding 0.35041</td>
<td>0.525</td>
<td>0.5757</td>
<td>0.6375</td>
<td>0.7246</td>
</tr>
<tr>
<td></td>
<td>Adversarial</td>
<td>0.65</td>
<td>0.69</td>
<td>0.718</td>
<td>0.724</td>
</tr>
</tbody>
</table>

Table 1: Comparison between LDP and Adversarial Representation Learning: Using our proposed framework we compare the utility of LDP and ARL across different values of the privacy parameter \( \epsilon \).

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 2: SSIM metric for reconstruction attack with varying \( R \) and \( \epsilon \)

**D Lipschitz Constant Estimation**

We use the mixed integer programming based algorithm LipMip by [13] for computing the local Lipschitz constant. Their technique allows exactly computing the local Lipschitz constant of a neural network with ReLU non-linearities. Their key idea is to estimate the supremal norm of the Jacobian of the neural network. Since ReLU networks do not allow for differentiability, LipMip uses clark jacobian to circumvent the issue and encode the optimization objective of obtaining local Lipschitz constant over a pre-defined neighborhood as a mixed integer programming problem. Their algorithm searches for feasible regions and minimizes the gap between lower and upper bound on the Lipschitz constant.

**E Experimental Details**

Our experimental setup operates in three stages - i) Embedder training, ii) Obfuscator training, and iii) Private inference. Our codebase is available [here](https://drive.google.com/drive/folders/1DpHhS9u-Mypp3TvmYiue7BKfustyKw2v?usp=sharing) for reproducibility. We will release the code and all trained models publicly after the reviews. For all our experiments we use PyTorch([24]) with Nvidia-GeForce GTX TITAN GPU. We use \( \beta \)-VAE with \( \beta = 5 \) for the design of the embedder.

1. **Embedder Training**: We use embedding dimension as 8 for MNIST and FMNIST dataset. For the UTKFace dataset, we use embedding size as 10. We use Adam optimizer([16]) with a constant learning rate of 0.001. The VAE architecture for MNIST and FMNIST dataset is composed of three fully connected layers with non-linear activations and dropout.

2. **Obfuscator Training**: For ARL, we use \( \alpha = 0.99 \), for noisy regularization, we use \( \sigma = 0.01 \) and for contrastive regularization, we use \( \lambda = 1.0 \) with a margin of 25. All of these regularizations are trained jointly using Adam optimizer([16]).

3. **Private Inference**: In the this stage we use LipMip([13]) which is built upon Gurobi Optimizer([9]) for solving the Mixed-Integer programming formulation of local lipschitz constant estimation. For the metrics, we use \( d_X \) as infinity norm and \( d_Y \) as \( \ell_1 \)-norm. For the privacy parameters, we use \( \delta = 0.05 \) and \( R = 0.5 \) for MNIST, \( R = 0.2 \) for FMNIST and \( R = 0.1 \) for UTKFace. The choice of different \( R \) was based on visualizing samples from the training set and evaluating how far similar looking samples lie in the embedding space.
Figure 1: Neighborhood for different image datasets. The center image (in red) is the reconstruction of the original image with nearby images sampled from the embedding. Note that there are multiple dimensions and we have illustrated interpolation for only one here.

Table 3: $\ell_1$ metric for reconstruction attack with varying $R$ and $\epsilon$

<table>
<thead>
<tr>
<th>$\ell_1$</th>
<th>MNIST</th>
<th>UTKFace</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon=1$</td>
<td>0.0775</td>
<td>0.0968</td>
</tr>
<tr>
<td>$\epsilon=2$</td>
<td>0.0642</td>
<td>0.078</td>
</tr>
<tr>
<td>$\epsilon=5$</td>
<td>0.0565</td>
<td>0.0634</td>
</tr>
<tr>
<td>$\epsilon=10$</td>
<td>0.0549</td>
<td>0.0569</td>
</tr>
</tbody>
</table>

Table 4: $\ell_2$ metric for reconstruction attack with varying $R$ and $\epsilon$

<table>
<thead>
<tr>
<th>$\ell_2$</th>
<th>MNIST</th>
<th>UTKFace</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon=1$</td>
<td>61.87</td>
<td>60.70</td>
</tr>
<tr>
<td>$\epsilon=2$</td>
<td>63.04</td>
<td>61.80</td>
</tr>
<tr>
<td>$\epsilon=5$</td>
<td>63.88</td>
<td>63.15</td>
</tr>
<tr>
<td>$\epsilon=10$</td>
<td>64.06</td>
<td>63.82</td>
</tr>
</tbody>
</table>

Table 5: PSNR metric for reconstruction attack with varying $R$ and $\epsilon$

<table>
<thead>
<tr>
<th>PSNR</th>
<th>MNIST</th>
<th>UTKFace</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon=1$</td>
<td>61.36</td>
<td>60.77</td>
</tr>
<tr>
<td>$\epsilon=2$</td>
<td>63.37</td>
<td>62.99</td>
</tr>
<tr>
<td>$\epsilon=5$</td>
<td>64.59</td>
<td>64.59</td>
</tr>
<tr>
<td>$\epsilon=10$</td>
<td>64.93</td>
<td>64.93</td>
</tr>
</tbody>
</table>
References


Towards fast computation of certified robustness for relu networks. In *International Conference 

[37] H. Zhao, J. Chi, Y. Tian, and G. J. Gordon. Trade-offs and guarantees of adversarial representa-
tion learning for information obfuscation. *Advances in Neural Information Processing Systems*, 

[38] H. Zhao, C. Dan, B. Aragam, T. S. Jaakkola, G. J. Gordon, and P. Ravikumar. Fundamental 