Bounding the Invertibility of Privacy-Preserving Instance Encoding Using Fisher Information

Kiwan Maeng
Penn State University
kvm6242@psu.edu

Chuan Guo
FAIR, Meta
chuanguo@meta.com

Sanjay Kariyappa
Georgia Institute of Technology
sanjaykariyappa@gatech.edu

G. Edward Suh
FAIR, Meta & Cornell University
edsuh@meta.com

Abstract

Privacy-preserving instance encoding aims to encode raw data into feature vectors without revealing their privacy-sensitive information. When designed properly, these encodings can be used for downstream ML applications such as training and inference with limited privacy risk. However, the vast majority of existing schemes do not theoretically justify that their encoding is non-invertible, and their privacy-enhancing properties are only validated empirically against a limited set of attacks. In this paper, we propose a theoretically-principled measure for the invertibility of instance encoding based on Fisher information that is broadly applicable to a wide range of popular encoders. We show that dFIL can be used to bound the invertibility of encodings both theoretically and empirically, providing an intuitive interpretation of the privacy of instance encoding.

1 Introduction

Machine learning (ML) applications often require access to privacy-sensitive data. A model that predicts diseases with x-ray scans requires access to patient’s x-ray images. Next-word prediction models require access to user’s text input that can contain sensitive information [23]. To enable ML applications on privacy-sensitive data, instance encoding ([6]; Figure 1, left) aims to encode data in a way such that it is possible to run useful ML tasks—such as model training and inference—on the encoded data while the privacy of the raw data is preserved. The concept is widespread under many different names: private data sharing [16,17,48,5,34,32,37], learnable encryption [35,73,71,70], split learning [63,55], split inference [40,11], and vertical federated learning (vFL; [74,59,45]) are all collaborative training/inference schemes that operate on (hopefully) privately-encoded user data.

Unfortunately, the vast majority of existing instance encoding schemes rely on heuristics rather than rigorous theoretical arguments to justify their privacy-enhancing properties. Most existing works [35,73,64,65,45] claimed that their schemes are non-invertible by simply testing them against certain input reconstruction attacks. However, these schemes may not be private under more carefully designed attacks; in fact, many encoding schemes that were initially thought to be private have been shown to be vulnerable over time [6,7]. While there exist instance encoding schemes that are based on a rigorous theory, e.g., metric differential privacy (metric-DP; [8,16,17,44]), their theoretical analysis is only applicable to very simple encoders and cannot be applied to more powerful encoders (e.g., DNN-based encoders) that are common in practice [73,64,65,45,8,54,53].
To fill the gap, we propose a theoretically-principled and easily applicable framework for quantifying the privacy of instance encoding by bounding its invertibility. Our framework uses (diagonal) Fisher information leakage (dFIL; [18, 22, 20])—an information-theoretic measure of privacy with similar properties to differential privacy (DP; [14, 15]). dFIL can be computed for common privacy-enhancing mechanisms and can lower-bound the expected mean squared error (MSE) of an input reconstruction attack. We apply this reasoning to instance encoding and show that dFIL can serve as a good measure for the invertibility of instance encoding, as it can bound the reconstruction MSE against any attacker that tries to reconstruct the private input from the encoding. dFIL can be easily calculated for any encoders that are differentiable and randomized—being applicable to most of the popular encoders with minimal modifications. To the best of our knowledge, our work is the first to theoretically lower-bound the invertibility of instance encoding and use it to design private training/inference systems.

Contributions  Our main contributions are as follows:

1. We adapt the result of prior works [18, 20] for instance encoding to show how dFIL can lower bound the MSE of particular input reconstruction attacks (i.e., unbiased attacks) that aim to reconstruct the raw data given the encoding. We show how popular encoders can be modified minimally for dFIL to be applied (Section 3.1).

2. We extend the result of prior works [18, 20] and show that dFIL can lower bound the MSE of any input reconstruction attack (e.g., strong attacks leveraging knowledge of the input prior; Section 3.2). Our extension involves a novel application of the classical van Trees inequality [62] and connecting it to the problem of score matching [57] in distribution estimation.

3. We evaluate the lower bound using different attacks and encoders, and show that dFIL can be used to interpret the invertibility of instance encoding both theoretically and empirically (Section 3.3).

4. We show how dFIL can be used as a practical privacy metric and guide the design of privacy-enhancing training/inference systems with instance encoding (Section 4). We show that it is possible to achieve both high (theoretically-justified) privacy and satisfactory utility.

2  Motivation and background

2.1  Instance encoding

Instance encoding refers to the general concept of encoding raw input \( x \) using an encoding function \( \text{Enc} \), such that private information contained in \( x \) cannot be inferred from its encoding \( e = \text{Enc}(x) \), while \( e \) maintains enough utility for the downstream ML tasks. The principle behind the privacy-enhancing property of instance encoding is that the function \( \text{Enc} \) is hard to invert. However, prior works generally justify this claim of non-invertibility based on heuristics rather than rigorous theoretical analysis [64, 65, 45], except for a few cases where the encoder is very simple [16, 17, 44].

Attacks against instance encoding  Given an instance encoder \( \text{Enc} \), the goal of a reconstruction attack is to recover its input. Formally, given \( e = \text{Enc}(x) \), an attack \( \text{Att} \) aims to reconstruct \( x \) from \( e \): \( \hat{x} = \text{Att}(e) \). Such an attack can be carried out in several ways. If \( \text{Enc} \) is known, \( x \) can be obtained by solving the following optimization [27]: \( \hat{x} = \arg \min_{\xi} ||e - \text{Enc}(\xi)||_2^2 \). This attack can be further improved when some prior of the input is known [49, 61]. For instance, images can be regularized with total variation (TV) prior to reduce high-frequency components [49]. Alternatively, if samples from the input distribution can be obtained, a DNN that generates \( \hat{x} \) from \( e \) can be trained [53, 27, 12].
Privacy metrics for instance encoding. Measuring how much information an observation leaks about a secret quantity it depends on is a classical question that has long been studied in information theory [38, 3, 13, 33], and instance encoding privacy is directly relevant to the stream of works. Prior works proposed metrics providing a strong notion of privacy (e.g., Arimoto’s mutual information [3] or Sibson’s mutual information [38]). However, it is challenging to apply these metrics to ML and instance encoding which use high-dimensional data [33], because these metrics have to be calculated and optimized over the prior of the dataset, the encodings, and/or their joint distribution, all of which are unknown and hard to accurately model [33].

Differential privacy (DP) [14, 15], one of the most popular frameworks to quantify ML privacy [1], is not suitable for instance encoding [6]. DP guarantees the worst-case indistinguishability of the encoding from two different inputs, which significantly damages the utility [6] (Appendix 7.5). Metric-DP [3]—a weaker notion of DP—is applicable to instance encoding and has been used by several prior works [16, 17, 44, 48, 37]. However, metric-DP is applicable only when the encoder is extremely simple. For example, prior works used pixel-level perturbation (e.g., pixelization, blurring [16, 17]) or a single-layer convolution [44] as an encoder. These extremely-simple encoders are in stark contrast with the complex encoders most practitioners use (e.g., split learning [54], split inference [65], and vFL [44, 59] all uses DNN-based encoders), and results in a significant utility loss (Section 4.2). Applying metric-DP to an arbitrary DNN-based encoder is NP-hard [66]. Unlike these works that focus on indistinguishability, our work directly quantifies non-invertibility, a weaker but still relevant definition of privacy (Section 3).

Due to the limitations in existing theoretical metrics, the vast majority of works in the field are relying on empirical measures to demonstrate their encoder’s privacy-enhancing properties. Some [46, 45] simply test their encoders against a limited set of attacks, while others [64, 65, 51] use heuristical privacy metrics without rigorous theoretical arguments. Using heuristical metrics that are not backed by a theoretical interpretation is dangerous, because a seemingly useful metric can actually severely mis-characterize the privacy of a system [38]. It is of both interest and practical importance to develop a privacy metric that is both theoretically-meaningful and widely applicable.

A concurrent work [72] proposed a notion of PAC privacy that can bound the adversary’s attack success rate in theory for instance encoding. PAC privacy bound has not yet been evaluated empirically under a realistic attack and use cases, which remains an interesting future work.

2.2 Fisher information leakage

Fisher information leakage (FIL; [18, 22, 20]) is a measure of leakage through a privacy-enhancing mechanism. Let \( M \) be a randomized mechanism on data sample \( x \), and let \( o \sim M(x) \) be its output. Suppose the log density function \( \log p(o; x) \) is differentiable w.r.t. \( x \) and satisfies the following regularity condition: \( E_o[\nabla_x \log p(o; x) | x] = 0 \). The Fisher information matrix (FIM) \( I_o(x) \) is:

\[
I_o(x) = E_o[\nabla_x \log p(o; x) \nabla_x \log p(o; x)^\top].
\]

Cramér-Rao bound Fisher information is a compelling privacy metric as it directly relates to the mean squared error (MSE) of a reconstruction adversary through the Cramér-Rao bound [41]. In detail, suppose that \( \hat{x}(o) \) is an unbiased estimate (or reconstruction) of \( x \) given the output of the randomized private mechanism \( o \sim M(x) \). Then:

\[
E_o[||\hat{x}(o) - x||^2_2 / d] \geq \frac{d}{\text{Tr}(I_o(x))},
\]

where \( d \) is the dimension of \( x \) and \( \text{Tr} \) is the trace of a matrix. Guo et al. [20] defined a scalar summary of the FIM called diagonal Fisher information leakage (dFIL):

\[
d\text{FIL}(x) = \frac{\text{Tr}(I_o(x))}{d},
\]

hence the MSE of an unbiased reconstruction is lower bounded by the reciprocal of dFIL. dFIL varies with \( x \), allowing it to reflect the fact that certain samples may be more vulnerable to reconstruction.

Limitations Although the Cramér-Rao bound gives a mathematically rigorous interpretation of dFIL, it depends crucially on the unbiasedness assumption, i.e., \( E_o[\hat{x}(o)] = x \). In practice, most real-world reconstruction attacks use either implicit or explicit priors about the data distribution and are biased (e.g., attacks using TV prior or a DNN). It is unclear how dFIL should be interpreted in these more realistic settings. In Section 3.2, we give an alternative theoretical interpretation based on the van Trees inequality [62], which lower-bounds the MSE of any adversary, biased or unbiased.
3 Quantifying the invertibility of an encoding

Motivated by the lack of theoretically-principled privacy metrics, we propose to adapt the Fisher information leakage framework to quantify the invertibility of instance encoding. We show that many existing encoders can be modified minimally to be interpreted with dFIL and the Cramér-Rao bound. Subsequently, we extend the framework and derive a novel bound for the reconstruction error of arbitrary attacks, by establishing a connection to van Trees inequality and score matching.

Threat model We focus on reconstruction attacks that aim to invert an encoding, i.e., reconstruct the input $x$ given its encoding $e$. We assume that the attacker has full knowledge of the encoder $\text{Enc}$ except for the source of randomness. We consider both unbiased attacks and biased attacks that can use arbitrary prior knowledge about the data distribution.

Privacy definition We consider $\text{Enc}$ to be private if $x$ cannot be reconstructed from the encoding $e$. We measure the reconstruction error with the mean squared error (MSE), defined as $||\hat{x} - x||^2/d$. Although MSE does not exactly indicate semantic similarity, it is widely applicable and used as a proxy for semantic similarity [69, 43]. Preventing low reconstruction MSE does not necessarily protect against other attacks (e.g., property inference [50]), which we leave as future work.

3.1 Fisher information leakage for instance encoding

To adapt the framework of Fisher information to the setting of instance encoding, we consider the encoding function $\text{Enc}$ as a privacy-enhancing mechanism (cf. $\mathcal{M}$ in Section 2.2) and use dFIL to measure the invertibility of the encoding $e = \text{Enc}(x)$. However, many instance encoders do not meet the regularity conditions in Section 2.2, making dFIL ill-defined. For example, popular DNN-based encoders do not produce randomized output, and their log density function $\log p(o; x)$ may not be differentiable when operators like ReLU or max pooling are present.

Fortunately, many popular encoders can meet the required conditions with small changes. For example, DNN-based encoders can be modified by (1) replacing any non-smooth functions with smooth functions (e.g., tanh or GELU [29] instead of ReLU, average pooling instead of max pooling), and (2) adding noise at the end of the encoder for randomness. Alternative modifications are also possible. In particular, if we add random Gaussian noise to the output of a differentiable, deterministic encoder $\text{Enc}_D$ (e.g., DNN): $\text{Enc}(x) = \text{Enc}_D(x) + \mathcal{N}(0, \sigma^2)$, the FIM of the encoder becomes [22]:

$$J_e(x) = \frac{1}{\sigma^2} J^\top_{\text{Enc}_D}(x) J_{\text{Enc}_D}(x),$$

where $J_{\text{Enc}_D}$ is the Jacobian of $\text{Enc}_D$ with respect to the input $x$ and can be easily computed using a single backward pass. Then, Equation 2 can be used to bound the reconstruction error, provided the attack is unbiased. Other popular encoders can be modified similarly.

3.2 Bounding the reconstruction of arbitrary attacks

Most realistic reconstruction attacks are biased, and their reconstruction MSE is not lower bounded by Equation 2. Consider an (biased) attacker who knows the mean $\mu$ of the input data distribution. If the attacker simply outputs $\mu$ as the reconstruction of any input $x$, the expected MSE will be the variance of the data distribution regardless of dFIL. The above example shows a crucial limitation of the Cramér-Rao bound interpretation of dFIL: it does not take into account any prior information the adversary has about the data distribution, which is abundant in the real world (Section 2.1).

Bayesian interpretation of Fisher information We adopt a Bayesian interpretation of dFIL as the difference between an attacker’s prior and posterior estimate of the input $x$. This is achieved through the classical van Trees inequality [62]. We state the van Trees inequality in Appendix 7.3 and use it to derive our MSE bound for arbitrary attacks below as a corollary; proof is in Appendix 7.4.

Corollary 1. Let $\pi$ be the input data distribution and let $f_\pi(x)$ denote its density function with respect to Lebesgue measure. Suppose that $\pi$ satisfies the regularity conditions of van Trees inequality (Theorem 3), and let $J(f_\pi) = \mathbb{E}_\pi \nabla_x \log f_\pi(x) \nabla_x \log f_\pi(x)\top$ denote the information theorist’s Fisher information [27] of $\pi$. For a private mechanism $\mathcal{M}$ and any reconstruction attack $\hat{x}(o)$
operating on $\mathbf{o} \sim \mathcal{M}(\mathbf{x})$:  
\[
\mathbb{E}_e\mathbb{E}[\|\mathbf{x} - \mathbf{\hat{x}}\|^2_2/d] \geq \frac{1}{\mathbb{E}_e[d\text{FIL}(\mathbf{x})] + \text{Tr}(\mathcal{J}(f_\pi))/d}.
\]  

**Implications of Corollary 1** We can readily apply Corollary 1 to instance encoding by replacing $\mathcal{M}$ with $\text{Enc}$ and $\mathbf{o}$ with $\mathbf{e}$ as in Section 3.1. Doing so leads to several interesting implications:

1. Corollary 1 is a population-level bound that takes expectation over $\mathbf{x} \sim \pi$. This is necessary because given any fixed sample $\mathbf{x}$, there is always an attack $\mathbf{x}(\mathbf{e}) = \mathbf{x}$ that perfectly reconstructs $\mathbf{x}$ without observing the encoding $\mathbf{e}$. Such an attack would fail in expectation over $\mathbf{x} \sim \pi$.

2. The term $\mathcal{J}(f_\pi)$ captures the prior knowledge about the input. When $\mathcal{J}(f_\pi) = 0$, the attacker has no prior information about $\mathbf{x}$, and Corollary 1 reduces to the unbiased bound in Equation 2. When $\mathcal{J}(f_\pi)$ is large, the bound becomes small regardless of $\mathbb{E}_e[d\text{FIL}(\mathbf{x})]$, indicating that the attacker can simply guess with the input prior and achieve a low MSE.

3. $d\text{FIL}$ can be interpreted as capturing how much easier reconstructing the input becomes after observing the encoding ($\mathbb{E}_e[d\text{FIL}(\mathbf{x})]$ term) as opposed to only having knowledge of the input distribution ($\text{Tr}(\mathcal{J}(f_\pi))/d$ term). Using such a change in the belief after an observation as a privacy metric is common in information theory \[38, 13\].

**Estimating $\mathcal{J}(f_\pi)$** The term $\mathcal{J}(f_\pi)$ captures the prior knowledge of the input and plays a crucial role in Corollary 1. In simple cases where $\pi$ is a known distribution whose density function follows a tractable form (e.g., when the input follows a Gaussian distribution), $\mathcal{J}(f_\pi)$ can be directly calculated. In such settings, Corollary 1 gives a meaningful theoretical lower bound for the reconstruction MSE. However, most real-world data distributions do not have a tractable form and $\mathcal{J}(f_\pi)$ must be estimated from data. Fortunately, the $\nabla_\mathbf{x} \log f_\pi(\mathbf{x})$ term in $\mathcal{J}(f_\pi)$ is a well-known quantity called the score function in generative AI literature, and there exists a class of algorithms known as score matching \[36, 47, 57\] that aim to estimate the score function from the data. We leverage these techniques to estimate $\mathcal{J}(f_\pi)$ when it cannot be calculated; details are in Appendix 7.1. As $\mathcal{J}(f_\pi)$ only depends on the dataset and not the encoder, its value can be estimated once for each dataset and shared across the community. The estimation of $\mathcal{J}(f_\pi)$ is expected to improve with the advancement of generative AI.

**Using Corollary 1 in practice** When $\mathcal{J}(f_\pi)$ is known (e.g., Gaussian), the bound from Corollary 1 always hold. However, when estimating $\mathcal{J}(f_\pi)$ from data, the bound can be incorrect due to several reasons, including improper modeling of the score function, not having enough representative samples, or the regularity conditions of the van Trees inequality not being met. The bound can also be loose when tightness conditions of the van Trees inequality do not hold. Even when the bound is not exact, however, Equations 2 and 3 can be interpreted to suggest that increasing $1/d\text{FIL}$ makes reconstruction harder. Thus, we argue that $d\text{FIL}$ still serves as a useful privacy metric that in theory bounds the invertibility of an instance encoding. When not exact, the bound should be viewed more as a guideline for interpreting and setting $d\text{FIL}$ in a data-dependent manner.

### 3.3 Evaluation of the bound

We show that Corollary 1 accurately reflects the reconstruction MSE on both (1) synthetic data with known $\mathcal{J}(f_\pi)$, and (2) real world data with estimated $\mathcal{J}(f_\pi)$.
3.3.1 Synthetic data with known $\mathcal{J}(f_\pi)$

Evaluation setup We consider a synthetic Gaussian input distribution: $x \sim \mathcal{N}(0, \tau^2 I_d)$ with $d = 784$ and $\tau = 0.05$. It can be shown that $\text{Tr}(\mathcal{J}(f_\pi))/d = 1/\tau^2$, hence a larger $\tau$ forces the data to spread out more and reduces the input prior. We use a simple encoder which randomly projects the data to a 10,000-dimensional space and then adds Gaussian noise, i.e., $e = Mx + \mathcal{N}(0, \sigma^2)$, where $M \in \mathbb{R}^{10,000 \times 784}$. We evaluate with a simple encoder to make the attack easier, but note that our bound should hold for more complex encoders (e.g., DNN-based) as well.

Attacks We evaluate our bound against two different attacks. An unbiased attack (Attack-ub) solves the following optimization: $x(e) = \arg \min_{x_0} ||e - \text{Enc}(x_0)||_2^2$. The attack is unbiased as the objective is convex, and $x$ is recovered in expectation. A more powerful biased attack (Attack-b) adds a regularizer term $\lambda \log p_\tau(x_0)$ to the above objective, where $p_\tau$ is the density function of $\mathcal{N}(0, \tau^2 I_d)$. One can show that with a suitable choice of $\lambda$, this attack returns the maximum a posteriori estimate of $x$, which leverages knowledge of the input distribution. Details are in Appendix 7.2.

Result Figure 3 (right) plots the MSE of the two attacks, and the bounds for unbiased (Equation 2) and arbitrary attack (Equation 5). The MSE of Attack-ub (red circle) matches the unbiased attack lower bound (Bound-ub; red dashed line), showing the predictive power of Equation 2 against this restricted class of attacks. Under Attack-b (blue triangle), however, Bound-ub breaks. Our new bound from Equation 5 (Bound-ours; blue dotted line) reliably holds for both attacks, initially being close to the unbiased bound and converging to guessing only with the input prior (attaining $\tau^2$).

3.3.2 Real world data with estimated $\mathcal{J}(f_\pi)$

Evaluation setup We also evaluated Corollary 1 on MNIST [10] and CIFAR-10 [42]. Here, we estimated $\mathcal{J}(f_\pi)$ using sliced score matching [57]. As discussed in Appendix 7.1, a moderate amount of randomized smoothing (adding Gaussian noise to the raw input [9]) is necessary to ensure that the score estimation is stable and that regularity conditions of the van Trees inequality are satisfied. We used a simple CNN-based encoder: $e = \text{Conv}(x) + \mathcal{N}(0, \sigma^2)$.

Attacks We evaluated Attack-ub, which is the same as in Section 3.3.1 and Attack-b, which is a trained DNN that outputs the reconstruction given an encoding [45]. We also evaluated regularization-based attacks [49, 61] and obtained similar results; we omit those results for brevity.
**Result**  Figures 2 and 3 plot the result with a randomized smoothing noise of $\mathcal{N}(0, 0.25^2)$. Again, Bound-ub correctly bounds the MSE achieved by Attack-ub. While Attack-b is not as effective for very low $1/dFIL$, it outperforms Attack-ub for high $1/dFIL$, breaking Bound-ub. In comparison, Corollary 1 estimated using score matching (Bound-ours) gives a valid lower bound for both attacks.

Figures 2 and 3 also highlights some of the reconstructions visually. Here, the left-hand side number indicates the target $1/dFIL$ and the images are reconstructed using Attack-b. In both figures, it can be seen that dFIL correlates well with the visual quality of reconstructed images, with higher values of $1/dFIL$ indicating less faithful reconstructions. See Appendix: Figure 10–11 for more results.

Figure 4 additionally shows the result with a much smaller randomized smoothing noise of $\mathcal{N}(0, 0.01^2)$. Unlike previous results, Bound-ours breaks around $1/dFIL=10^{-3}$. We suspect it is due to score matching failing when the data lie on a low-dimensional manifold and the likelihood changes rapidly near the manifold boundary, which can be the case when the smoothing noise is small. Nonetheless, the bound still correlates well with MSE and the visual reconstruction quality. We claim that dFIL still serves as a useful privacy metric, with a theoretically-principled interpretation and a strong empirical correlation to invertibility. More reconstructions are in Appendix: Figure 12.

4 Case study 1: split inference with dFIL

4.1 Private split inference with dFIL

Split inference [40, 4, 65] is a method to run inference of a large DNN that is hosted on the server, without the client disclosing raw input. It is done by running the first few layers of a large DNN on the client device and sending the intermediate activation, instead of raw data, to the server to complete the inference. The client computation can be viewed as instance encoding, where the first few layers on the client device act as an encoder. However, without additional intervention, split inference by itself is not private because the encoding can be inverted [27].

We design a private split inference system by measuring and controlling the invertibility of the encoder with dFIL. Because the encoder of split inference is a DNN, dFIL can be calculated using Equation 4 with minor modifications to the network (Section 3.1).

**Optimizations**  There are several optimizations to improve the model accuracy for the same dFIL: (1) We calculate the amount of noise that needs to be added to the encoding to achieve a target dFIL, and add a similar amount of noise during training. (2) For CNNs, we add a compression layer—a convolution layer that reduces the channel dimension significantly—at the end of the encoder and a corresponding decompression layer at the beginning of the server-side model. Similar heuristics were explored in prior works [11] [45]. (3) We add an SNR regularizer that is designed to maximize the signal-to-noise ratio of the encoding. From Equations 3–4, the noise that needs to be added to achieve a certain dFIL is $\sigma = \sqrt{\frac{\text{Tr}(J_{\text{Enc,D}}(x)J_{\text{Enc,D}}(x))}{d\ast dFIL}}$. Thus, maximizing the signal-to-noise ratio (SNR) of the encoding ($e^T e/\sigma^2$) is equivalent to minimizing $\text{Tr}(J_{\text{Enc,D}}(x)J_{\text{Enc,D}}(x))$, which we add to the optimizer during training. These optimizations were selected from comparing multiple heuristics from prior work [60] [28] [46] [65] [45] [11], and result in a notable improvement of test accuracy.

4.2 Evaluation of dFIL-based split inference

4.2.1 Evaluation setup


**Detailed setups and attacks**  For ResNet-18, we explored three split inference configurations: splitting early (after the first convolution layer), in the middle (after block 4), and late (after block 6). We evaluated the empirical privacy with a DNN attacker [45] and measured the reconstruction quality with structural similarity index measure (SSIM) [31]. Other popular attacks showed similar trends (see Appendix: Figure 19). NCF-MLP translates a user id (uid) and a movie id (mid) into embeddings.
with an embedding table and sends them through a DNN to make a prediction. We split the NCF-MLP model after the first linear layer of the MLP and tried reconstructing the original uid and mid from the encoding. This is done by first reconstructing the embeddings from the encoding using direct optimization \((\hat{\text{emb}}(e) = \arg \min_{\text{emb}} || e - \text{Enc}(\text{emb}_0)||^2_2)\), and finding the original uid and mid by choosing the closest embedding value in the embedding table: \(id = \arg \min_i ||\hat{\text{emb}} - \text{Emb}[i]||^2_2\), where \(\text{Emb}[i]\) is the \(i\)-th entry of the embedding table. For DistilBert, we again explored three different splitting configurations: splitting early (right after block 0), in the middle (after block 2), and late (after block 4). We use a similar attack to NCF-NLP to retrieve each word token.

### 4.2.2 Evaluation results

#### Privacy

Figure 5 shows the attack result for ResNet-18 and CIFAR-10. Setups with lower dFIL lead to lower SSIM (left) and less identifiable images (right), indicating that dFIL strongly correlates with the attack success rate. The figures also show that the privacy leakage estimated by dFIL can be conservative. Some setups show empirically-high privacy even when dFIL indicates otherwise, especially when splitting late. See Appendix: Figure 13 for more results. Figures 6 show the attack result for NCF-MLP (left) and DistilBert (right). Setups with lower dFIL again consistently showed a worse attack success rate. A sample reconstruction for DistilBert is shown in Appendix: Table 5.

#### Utility

Table 1 summarizes the test accuracy of the split inference models, where \(1/dFIL\) is chosen so that the attacker’s reconstruction error is relatively high. For the same \(1/dFIL\), our proposed optimizations (Ours) improve the accuracy significantly compared to simply adding noise (No opt.). In general, reasonable accuracy can be achieved with encoders with relatively low dFIL. Accuracy degrades more when splitting earlier, indicating that using a DNN encoder with many layers can benefit utility. Prior works empirically observed a similar trend where splitting earlier degrades the privacy-utility tradeoff, because the encoding becomes leakier [49]. Our framework supports complex DNN encoders, which is a distinguishing benefit compared to prior theoretical works [16,17,44].

### 5 Case study 2: training with dFIL

#### 5.1 Training on encoded data with dFIL

We consider a scenario where users publish their encoded private data, and a downstream model is trained on the encoded data. We use the first few layers of a pretrained model as the encoder by
Table 1: Test accuracy for different split inference setups with different dFIL. Base accuracy in parenthesis.

<table>
<thead>
<tr>
<th>Setup</th>
<th>Split</th>
<th>dFIL</th>
<th>No opt.</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10 + ResNet-18</td>
<td>early</td>
<td>10</td>
<td>10.70%</td>
<td>74.44%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>12.94%</td>
<td>91.35%</td>
</tr>
<tr>
<td></td>
<td>middle</td>
<td>10</td>
<td>33.54%</td>
<td>84.27%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>33.54%</td>
<td>84.27%</td>
</tr>
<tr>
<td></td>
<td>late</td>
<td>10</td>
<td>78.48%</td>
<td>92.35%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>78.48%</td>
<td>92.35%</td>
</tr>
<tr>
<td>CIFAR-100 + ResNet-18</td>
<td>early</td>
<td>10</td>
<td>1.00%</td>
<td>44.10%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>1.00%</td>
<td>40.69%</td>
</tr>
<tr>
<td></td>
<td>middle</td>
<td>10</td>
<td>1.98%</td>
<td>59.51%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>1.29%</td>
<td>41.29%</td>
</tr>
<tr>
<td></td>
<td>late</td>
<td>10</td>
<td>8.63%</td>
<td>65.77%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>2.01%</td>
<td>39.18%</td>
</tr>
<tr>
<td>MovieLens-20M + NCF-MLP (AUC: 0.8228)</td>
<td>early</td>
<td>1</td>
<td>0.8172</td>
<td>0.8286</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0.7459</td>
<td>0.8251</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>0.6120</td>
<td>0.8081</td>
</tr>
<tr>
<td>GLUE-SST2 + DistilBert (acc: 91.04%)</td>
<td>early</td>
<td>10</td>
<td>50.80%</td>
<td>82.80%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>49.08%</td>
<td>81.88%</td>
</tr>
<tr>
<td></td>
<td>middle</td>
<td>10</td>
<td>76.61%</td>
<td>83.03%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>61.93%</td>
<td>82.22%</td>
</tr>
<tr>
<td></td>
<td>late</td>
<td>10</td>
<td>90.25%</td>
<td>83.03%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>82.68%</td>
<td>82.82%</td>
</tr>
</tbody>
</table>

Table 2: Accuracy from training with different encoders.

<table>
<thead>
<tr>
<th>Pretrain dataset</th>
<th>dFIL</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyImageNet</td>
<td>10</td>
<td>75.46%</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>42.57%</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>10</td>
<td>80.16%</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>70.27%</td>
</tr>
<tr>
<td>CIFAR-10 (held-out 20%)</td>
<td>10</td>
<td>81.99%</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>78.65%</td>
</tr>
</tbody>
</table>

freezing the weights and applying the necessary changes in Section 3.1. Then, we use the rest of the model with its last layer modified for the downstream task and finetune it with the encoded data. We found that similar optimizations from split inference benefit this use case as well.

5.2 Evaluation of dFIL-based training

We evaluate the model utility and show that it can reach a reasonable accuracy when trained on encoded data. We omit the privacy evaluation as it is similar to Section 4.2.2.

Evaluation setup We train a ResNet-18 model for CIFAR-10 classification. First, we train the model on one of three different datasets: (1) TinyImageNet [58], (2) CIFAR-100 [42], and (3) held-out 20% of CIFAR-10. Then, layers up to block 4 are frozen and used as the encoder. The CIFAR-10 training set is encoded using the encoder and used to finetune the rest of the model. The setup mimics a scenario where some publicly-available data whose distribution is similar (TinyImageNet, CIFAR-100) or the same (held-out CIFAR-10) with the target data is available and is used for encoder training. Detailed hyperparameters are in Appendix 7.2.

Evaluation result Table 2 summarizes the result. Our design was able to achieve a decent accuracy using encoded data with relatively safe $\frac{1}{dFIL}$ values (10–100). The result indicates that model training with privately encoded data is possible. The achieved accuracy was higher when the encoder was trained with data whose distribution is more similar to the downstream task (CIFAR-10). We believe more studies in hyperparameter/architecture search will improve the result.

6 Limitations

We showed that dFIL can theoretically quantify the invertibility of instance encoding by providing a reconstruction error bound against arbitrary attackers. We subsequently showed that dFIL can be used to guide private training/inference systems that uses instance encoding. dFIL has several potential limitations, which needs to be carefully considered before being used:
1. Corollary 1 only bounds the MSE, which might not always correlate well with the semantic quality. To address this, van Trees inequality can be extended to an absolutely continuous function $\psi(x)$ to bound $E[||\psi(\hat{x}) - \psi(x)||_2^2/d]$ [19], which may be used to extend to metrics other than MSE.

2. Equation 5 provides an average case bound, so some data may experience lower reconstruction MSE than the bound. Similar average case bounds have also been used in prior works [13] [21]. When bounding invertibility, it is unclear how to define the privacy attack game to target worst-case reconstruction bound: for any sample, there exists a trivial attacker that always outputs that fixed sample from the distribution, which gives a worst-case MSE of zero for that sample. Leaker data that might experience lower reconstruction MSE than the bound can be potentially detected and handled by dynamically calculating dFIL for each sample. See Appendix: Figure [14].

3. For data types where MSE is not directly meaningful or the bound is inaccurate, it may not be straightforward to interpret the privacy of an encoding given its dFIL. In such cases, acceptable values of dFIL should be determined for each application through further research. The situation is similar to DP, where it is often not straightforward what privacy parameters (e.g., $\epsilon$, $\delta$) need to be used [39].

4. Systems with the same dFIL may actually have different invertibility, as the bound from dFIL may be conservative. Comparing the privacy of two different systems using dFIL should be done with caution because dFIL is a lower bound rather than an accurate measure of invertibility.

References


7 Appendix

7.1 Score matching details

We found that using score matching \cite{57} does not work reliably when the data’s structure lies on a low-dimensional manifold (e.g., natural images). We found that applying randomized smoothing \cite{9}, which adds Gaussian noise to the image for robust training, helps stabilize score matching as it smoothens the density function. Randomized smoothing also makes the bound tighter. We observed that adding a reasonable amount of noise (e.g., standard deviation of 0.25, which was originally used by Cohen et al. \cite{9}) works well in general, but adding only small noise (standard deviation of 0.01) does not. We show both results in Section 3.3.1.

7.2 Hyperparameters

**Attacks** For attacks in Section 3.3.1, 3.3.2, and 4.2, we used the following hyperparameters. For the optimizer-based attack for Gaussian synthetic input, we used Adam with $\text{lr}=10^{-3}$, and $\lambda=0.1$–100 for the regularizer. For the optimizer-based attack for NCF-MLP and DistilBert, we used Adam with $\text{lr}=0.1$. For the DNN-based attack for MNIST and CIFAR-10 (Figure 2, 3, 5), we used a modified DNN from Li et al. \cite{45}, which uses a series of convolution (Conv) and convolution transpose (ConvT) layers interspersed with leaky ReLU of slope 0.2. All the models were trained for 100 epochs using Adam with $\text{lr}=10^{-3}$. Below summarizes the architecture parameters. For DNN-based attacks in Section 3.3.2, we put a sigmoid at the end. For the attack in Section 4.2, we do not.

<table>
<thead>
<tr>
<th>Dataset + encoder Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST + Conv $3\times\text{Conv}(c_{out}=16, k=3, s=1) + \text{ConvT}(c_{out}=32, k=3, s=1, op=0)$ $+ \text{ConvT}(c_{out}=1, k=3, s=1, op=0)$</td>
</tr>
<tr>
<td>CIFAR-10 + split-early $3\times\text{Conv}(c_{out}=64, k=3, s=1) + \text{ConvT}(c_{out}=128, k=3, s=1, op=0)$ $+ \text{ConvT}(c_{out}=3, k=3, s=1, op=0)$</td>
</tr>
<tr>
<td>CIFAR-10 + split-middle $3\times\text{Conv}(c_{out}=128, k=3, s=1) + \text{ConvT}(c_{out}=128, k=3, s=2, op=1)$ $+ \text{ConvT}(c_{out}=3, k=3, s=2, op=1)$</td>
</tr>
<tr>
<td>CIFAR-10 + split-late $3\times\text{Conv}(c_{out}=256, k=3, s=1) + 2\times\text{ConvT}(c_{out}=256, k=3, s=2, op=1)$ $+ \text{ConvT}(c_{out}=3, k=3, s=2, op=1)$</td>
</tr>
</tbody>
</table>

**Split inference** Below are the hyperparameters for the models used in Section 4.2. For ResNet-18, we used an implementation tuned for CIFAR-10 dataset from \cite{54}, with ReLU replaced with GELU and max pooling replaced with average pooling. We used the default hyperparameters from the repository except for the following: $\text{bs}=128$, $\text{lr}=0.1$, and $\text{weight\_decay}=5 \times 10^{-4}$. For NCF-MLP, we used an embedding dimension of 32 and MLP layers of output size $[64, 32, 16, 1]$. We trained NCF-MLP with Nesterov SGD with momentum=0.9, $\text{lr}=0.1$, and batch size of 128 for a single epoch. We assumed 5-star ratings as click and others as non-click. For DistilBert, we used Adam optimizer with a batch size of 16, $\text{lr}=2 \times 10^{-5}$, $\beta_1=0.9$, $\beta_2=0.999$, and $\epsilon=10^{-8}$. We swept the compression layer channel dimension among 2, 4, 8, 16, and the SNR regularizer $\lambda$ between $10^{-3}$ and 100.

**Training** Below are the hyperparameters for the models evaluated in Section 5.2. We used the same model and hyperparameters with split inference for training the encoder with the pretraining dataset. Then, we freeze the layers up to block 4 and trained the rest for 10 epochs with CIFAR-10, with $\text{lr}=10^{-3}$ and keeping other hyperparameters the same.

**Execution Environment** All the evaluation was done on a single A100 GPU. The training and evaluation of each model ranged roughly from less than an hour (ResNet-18 with split-early, NCF-MLP) to 3–7 hours (ResNet-18 with split-late, DistilBert).

7.3 van Trees inequality

Below, we restate the van Trees Inequality \cite{19}, which we use to prove Theorem 5.
Theorem 2 (Multivariate van Trees inequality). Let \((\mathcal{X}, \mathcal{F}, P_\theta : \theta \in \Theta)\) be a family of distributions on a sample space \(\mathcal{X}\) dominated by \(\mu\). Let \(p(x|\theta)\) denote the density of \(X \sim P_\theta\) and \(I_\mathcal{X}(\theta)\) denotes its FIM. Let \(\theta \in \Theta\) follows a probability distribution \(\pi\) with a density \(\lambda_\pi(\theta)\) with respect to Lebesgue measure. Suppose that \(\lambda_\pi\) and \(p(x|\theta)\) are absolutely \(\mu\)-almost surely continuous and \(\lambda_\pi\) converges to \(\theta\) and the endpoints of \(\Theta\). Let \(\psi\) be an absolutely continuous function of \(\theta\) with the same output dimension of \(\theta\), and \(\psi_n\), an arbitrary estimator of \(\psi(\theta)\). Assume regularity conditions from Section 2.2 is met. If we make \(n\) observations \(\{x_1, x_2, \ldots, x_n\}\), then:

\[
\int_\Theta \mathbb{E}_\theta[|\psi_n - \psi(\theta)||^2_2] \lambda_\pi(\theta)\,d\theta \geq \frac{(\int \text{div}(\psi(\theta))\lambda_\pi(\theta)\,d\theta)^2}{n \int \text{Tr}(I_\mathcal{X}(\theta))\lambda_\pi(\theta)\,d\theta + \text{Tr}(\mathcal{J}(\lambda_\pi))},
\]

where \(\text{div}(\psi(\theta)) = \sum_i \frac{\partial \psi_i(\theta)}{\partial \theta_i}\).

7.4 Proof of Corollary 1

Proof. Let \(\psi\) be an identity transformation \(\psi(\theta) = \theta\). For the setup in Corollary 1, \(n = 1\) and \(\text{div}(\psi(\theta)) = d\), so the multivariate van Trees inequality from Theorem 2 reduces to:

\[
\mathbb{E}_n \mathbb{E}_\theta[||\hat{x} - x||^2_2/d] \geq \frac{d}{\mathbb{E}_n[\text{Tr}(I_n(x))] + \text{Tr}(\mathcal{J}(f_\pi))} = \frac{1}{\mathbb{E}_n[\text{dFIL}(x)] + \text{Tr}(\mathcal{J}(f_\pi))/d}
\]

7.5 Comparison with differential privacy

Differential privacy [1] is not well-suited for instance encoding, as we discuss in Section 2.1. We formulate and compare a DP-based instance encoding and compare it with our dFIL-based instance encoding in a split inference setup (Section 4) to show that DP-based instance encoding indeed does not work well.

To formulate DP for instance encoding, we define an adjacent set \(\mathcal{D}\) and \(\mathcal{D}'\) as two differing inputs. A randomized method \(\mathcal{A}\) is \((\alpha, \epsilon)\)-Rényi differentially private (RDP) if \(D_\alpha(\mathcal{A}(\mathcal{D})||\mathcal{A}(\mathcal{D}')) \leq \epsilon\) for \(D_\alpha(P||Q) = \frac{1}{\alpha-1} \log \mathbb{E}_P[Q(P(X)^{\alpha})]\). As DP provides a different privacy guarantee with dFIL, we use the theorem from Guo et al. [20] to derive an MSE lower bound using DP’s privacy metric for an unbiased attacker. Assuming a reconstruction attack \(\hat{x} = \hat{\mathcal{A}}(e)\) that reconstructs \(x\) from the encoding \(e = \text{Enc}(x)\), repurposing the theorem Guo et al. [20] gives:

\[
\mathbb{E}[||\hat{x} - x||^2_2/d] \geq \frac{\sum_{i=1}^d \text{diam}_i(\mathcal{X})^2/4d}{e^\epsilon - 1} \tag{6}
\]

for a \((2, \epsilon)\)-RDP Enc, where \(\mathcal{X}\) is the input data space. We can construct a \((2, \epsilon)\)-RDP encoder \(\text{Enc}_{RDP}\) from a deterministic encoder \(\text{Enc}_{DP}\) by scaling and clipping the encoding adding Gaussian noise, or \(\text{Enc}_{RDP} = \text{Enc}_{DP}(x)/\max(1, ||\text{Enc}_{DP}(x)||_2) + \mathcal{N}(0, \sigma^2)\), similarly to [1]. The noise to be added is \(\sigma = (2\epsilon)^2/\mathcal{E}^2\). Equation 6 for DP is comparable to Equation 2 for dFIL, and we use the two equations to compare DP and dFIL parameters. We use Equation 2 because 20 does not discuss the bound against biased attackers.

We evaluate both encoders for split inference using CIFAR-10 dataset and ResNet-18. We split the model after block 4 (split-middle from Section 4.2.1) and did not add any optimizations discussed in Section 4 for simplicity. For the DP-based encoder, we retrain the encoder with scaling and clipping so that the baseline accuracy without noise does not degrade. We ran both models without standardizing the input, which makes \(\text{diam}_i(\mathcal{X}) = 1\) for all \(i\).

Table 3 compares the test accuracy achieved when targeting the same MSE bound for an unbiased attacker using dFIL and DP, respectively. The result clearly shows that DP degrades the accuracy much more for similar privacy levels (same unbiased MSE bound), becoming impractical very quickly. DP suffers from low utility because DP is agnostic with the input and the model, assuming a
Table 4: Test accuracy when targeting the same MSE bound.

<table>
<thead>
<tr>
<th>Unbiased MSE bound</th>
<th>1e-5</th>
<th>1e-4</th>
<th>1e-3</th>
<th>1e-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>dFIL-based</td>
<td>93.09%</td>
<td>93.11%</td>
<td>92.52%</td>
<td>87.52%</td>
</tr>
<tr>
<td>DP-based</td>
<td>64.64%</td>
<td>56.68%</td>
<td>46.46%</td>
<td>33%</td>
</tr>
</tbody>
</table>

worst-case input and model weights. Our dFIL-based bound uses the information of the input and model weights in its calculation of the bound and can get a tighter bound.

7.6 Attack based on a diffusion model

We additionally designed a powerful, diffusion model-based reconstruction attacker to study the privacy of dFIL against the best-effort attacker, motivated from the fact that recently developed diffusion models [30] are excellent denoisers. During training of a diffusion model, (1) a particularly-designed noise is added to an image, and (2) a DNN is trained to predict and remove the noise [30]. The first part can be thought of as an instance encoder (that is purposely made easy to invert), and we can calculate its dFIL. The second part can be thought of as a reconstruction attacker. As the noising and denoising are specifically designed for the denoising to work well, we expect a mature, pretrained diffusion model to give a very good attack quality. We used DDPM [30] pretrained with CIFAR-10 from Google [67].

Figure 7 shows the result. The first column of each row shows the original image, and other columns show the reconstruction of our DDPM-based attacker with different dFIL. We scale and show dFIL with the same scale as Figure 5 as DDPM works with a different normalized image that produces dFIL at a different scale. Our new attack provided an interesting result: the attack was able to reconstruct conceptually similar images with the original image even when pixel-by-pixel reconstruction was prohibited by high $1/dFIL$. For example, 7th row at $1/dFIL = 49.5$ (6th column) successfully reconstructed a white car with a red background, although the reconstruction MSE was high and the design of the reconstructed car was nothing like the original image. The result shows that high-level information of the image (e.g., the color of the car/background, the orientation of the car, etc.) can still be preserved after encoding with a relatively high $1/dFIL$, which is why it is possible to perform downstream training/inference with a privately-encoded data without revealing the original data.

7.7 Per-pixel FIL

Instead of dFIL which is the average of the Fisher information leakage (FIL) for all the features (pixels), we can directly inspect the per-feature FIL to see which features (pixels) leak more information and can be reconstructed more easily. Figure 8 plots the per-pixel FIL from the split-middle setup of ResNet-18 and CIFAR-10 from Section 4.2.2. The result shows that identifying features of an object, such as the contour of a frog (8th column) or the tire of the car (7th column), are more easily leaked, following our intuition.

7.8 Additional figures and tables

Table 5: The reconstruction quality of an input is highly correlated with dFIL. Correct parts in bold.

<table>
<thead>
<tr>
<th>$1/dFIL$</th>
<th>Reconstructed text (from split-early)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-5}$</td>
<td>it’s a charming and often affecting journey.</td>
</tr>
<tr>
<td>1</td>
<td>it’s cones charming bound</td>
</tr>
<tr>
<td></td>
<td>often affecting journey closure</td>
</tr>
<tr>
<td>10</td>
<td>grounds yuki cum sign</td>
</tr>
<tr>
<td></td>
<td>recklessound fanuche pm stunt</td>
</tr>
</tbody>
</table>
Figure 7: Results from DDPM [30]-based reconstruction attack. High $1/dFIL$ prevents exact pixel-to-pixel reconstruction, but images that share some high-level features with the original image can be generated unless $1/dFIL$ is not too high.

Figure 8: Per-pixel FIL from the ResNet-18 + CIFAR-10 split-middle setup in Section 4.2.2. Image best viewed with colors.
Figure 9: Optimizer-based attack with total variation (TV) prior \[49\] against our split inference system in Section 4. The trend is very similar to Figure 5.

Figure 10: More reconstruction result of Figure 2.
Figure 11: More reconstruction result of Figure 3.

Figure 12: More reconstruction result of Figure 4.
Figure 13: Ten reconstructions with the best and worst SSIM values for various setups. The result is an extension of Figure 5. Images with a simple shape and high color contrast tend to be reconstructed more easily, which matches our intuition. Omitting results from $1/dFIL = 100$, as no meaningful reconstruction was possible.

Figure 14: Ten images with the lowest and highest dFIL values, for split-middle setup in Figure 5. Images with high dFIL tend to have a simpler shape and a high color contrast, potentially being easier to reconstruct. Individual dFIL value of each samples can potentially be used to detect data that are more leaky.