Exploiting Internal Randomness for Privacy in Vertical Federated Learning

Yulian Sun[1,3,⋆][0009−0007−0088−0385], Li Duan[1,2,⋆][0009−0001−1663−2622], Ricardo Mendes[1], Derui Zhu[1,4], Yue Xia[1,4], Yong Li[1,✉], and Asja Fischer[3]

[1] Huawei Technologies Düsseldorf, Germany
{yualian.sun1, li.duan, ricardo.mendes1, derui.zhu, yue.xia1, yong.li1}@huawei.com

[2] Paderborn University, Germany
liduan@mail.upb.de

[3] Ruhr University Bochum, Germany
yulian.sun@edu.ruhr-uni-bochum.de, asja.fischer@rub.com

[4] Technical University Munich, Germany
{derui.zhu, yue1.xia}@tum.de

Abstract. Vertical Federated Learning (VFL) is becoming a standard collaborative learning paradigm with various practical applications. Randomness is essential to enhancing privacy in VFL, but introducing too much external randomness often leads to an intolerable performance loss. Instead, as it was demonstrated for other federated learning settings, leveraging internal randomness —as provided by variational autoencoders (VAEs) —can be beneficial. However, the resulting privacy has never been quantified so far nor has the approach been investigated for VFL.

We therefore propose a novel differential privacy estimate, denoted as distance-based empirical local differential privacy (dELDP). It allows to empirically bound DP parameters of concrete components, quantifying the internal randomness with appropriate distance and sensitivity metrics. We apply dELDP to investigate the DP of VAEs and observe values up to $\epsilon \approx 6.4$ and $\delta = 2^{-32}$. Moreover, to link the dELDP parameters to the privacy of full VAE-includnig VFL systems in practice, we conduct comprehensive experiments on the robustness against sate-of-the-art privacy attacks. The results illustrate that the VAE system is effective against feature reconstruction attacks and outperforms other privacy-enhancing methods for VFL, especially when the adversary holds 75% of features in label inference attack.

Keywords: privacy · vertical federated learning · distance-based empirical differential privacy · variational autoencoder

⋆ Equal contribution

1 Introduction

Vertical Federated Learning (VFL) is a distributed learning paradigm that allows different participants to collaboratively train a joint model without sharing their raw data. In VFL, *different* features of each sample are held by different participants. A typical VFL model consists of multiple *bottom models* and one *top model* [37]. In principle, the participants feed their own features to the bottom models, whereas those with labels hold the top model(s) and coordinate the updates. VFL has been widely deployed in commercial services spanning various industrial sectors, and it is seen as prominent by leading AI players, including Google, Amazon, and Huawei, as VFL promotes data-based collaborations [18]. An overview of VFL workflow is shown in Figure 1.

Despite the remarkable success of VFL in real-world applications, there are increasing concerns about the vulnerabilities, especially input and model leakage of the basic VFL architecture. Besides attacking the VFL training [25, 31], an adversary can also extract information during the inference phase if a trained VFL model is released as a public service, threatening the input features of other participants and even the entire VFL model [25, 29]. For instance, Luo *et al.* [26] show how to utilize the model output to approximate the input features. Geng *et al.* [15] illustrate how the released model outputs can help an adversary reconstruct the entire model by employing transfer learning.

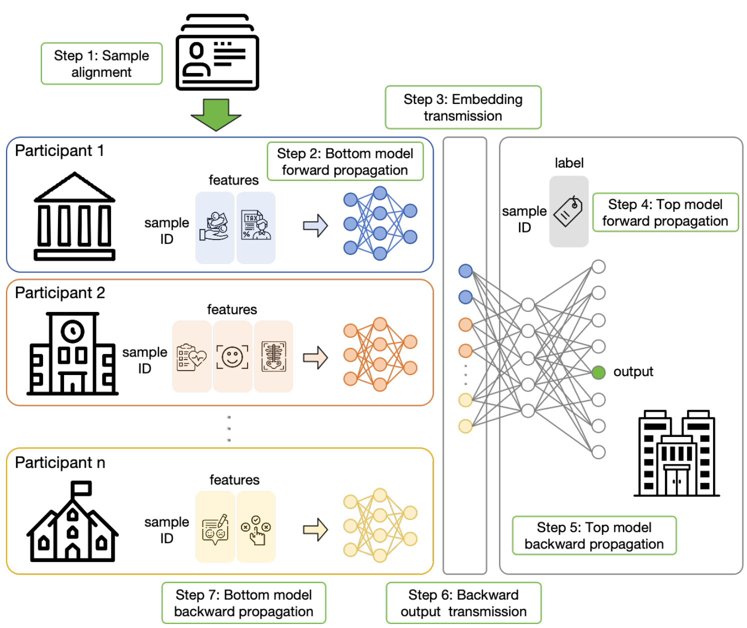

Fig. 1: VFL, the general workflow

1.1 Randomness for Privacy in VFL

Technically, enhancing privacy guarantee means processing the input data or intermediate states in VFL with randomness. The sources of randomness are (1) external sources, such as a random number generator, and (2) the internal structure combined with the distribution of input data.

Cryptography and classical differential privacy (DP) use external randomness. For example, a symmetric encryption scheme, such as AES-GCM [30], must use initialization vectors (IV) and keys that are (pseudo-)random and independent of the plaintext message, and the randomized response mechanism [12] needs input-independent random coins to decide whether a real or a random answer is given back. Quantifying privacy from external randomness is rather straightforward. For example, the privacy (security) guarantee of symmetric encryption algorithms is linear in its key and IV length [3], i.e., linear in the Shannon entropy of the external randomness. Other typical examples are the Laplace and Gaussian mechanisms in DP, where predefined b or σ can determine the privacy parameters (ϵ, δ).

However, the major downside of external randomness is also notable. hinder the utility [16], or the users may be concerned with the trustworthiness of centralized noise adding mechanisms. For instance, Liu *et al.* [25] apply homomorphic encryption (HE) to aggregate outputs from each bottom model for the secrecy of each individual embedding. However, HE can lead to an inflation of data transfer volume by 150×. Geng *et al.* [15] introduce noises into the output using a trusted third party (TTP) throughout training and inference. Nevertheless, TTP is usually considered an unjustified strong assumption in real-world scenarios.

In contrast, taking advantage of structural and data dependent randomness can lead to more compact and efficient solutions. This is because the structures such as variational autoencoder [19], global attention [10], and U-noise [21] are already parts of the machine learning model, i.e., the randomness is *internal*. Solutions such as PRECODE [31] and GATN [10] have demonstrated the merit of using internal randomness for privacy. On the other hand, quantifying the privacy guarantee of these randomness is far more challenging than in the external case. First, the structural randomness may not have an analytic form [21]. If the query structure spans several linear and non-linear layers, directly deriving a generic analytic form for the random distribution is intractable [19]. Second, the extracted privacy parameters may not be comparable or compatible with other mechanisms. For example, as pointed out by Burchard *et al.* [8], Duan's noise-less privacy [11] cannot be combined with DP directly.

1.2 Contributions and Paper Outline

Our proposal paves another way for exploiting the internal randomness for the privacy guarantee and we make the following contributions.

1. We propose a new differential privacy estimate called distance-based Empirical Local Differential Privacy (dELDP) that forms a good foundation for *em-*

pirical estimation of internal randomness in VFL. When parameterized with appropriate distance metric, sensitivity and estimators, the dELDP privacy parameters can be efficiently estimated for concrete structural parameters, trained model, and data sets.

2. We demonstrate the applicability of dELDP by estimating (ϵ, δ) of trained VAEs. Randomly initialized VAEs are integrated into existing VFL architecture, and the whole model is then trained with the original optimization goal and methods. We denote the VFL pipeline extended with VAE as SAIR, as it SAves the cost with Internal Randomness. Besides quantifiable privacy guarantee in the sense of dELDP, we also show that SAIR can enjoy minimal performance loss and be empirically composed with other privacy-enhancing technologies for improved resilience against attacks.
3. To link dELDP parameters to concrete privacy guarantee in practice, we also conduct comprehensive experiments against state-of-the-art attacks in VFL inference. Besides showing effective defences against feature reconstruction attacks, the results also illustrate that SAIR outperforms other privacy-preserving methods in VFL, especially when the adversary holds more than 50% of features in label inference attacks and model stealing attacks.

Outline. After the introduction of background knowledge and the survey of related work in Section 2, Section 3 elaborates dELDP definitions and its application. Section 4 and Section 5 contain detailed implementation, experiment description and highlights of empirical results for privacy and performance.

2 Background and Related Work

Distance-based Local Differential Privacy The main idea of Local differential privacy (LDP) is to perturb individual data samples with noises, and the usage of LDP is frequently seen in FL. Truex *et al.* [34] present LDP-Fed, a federated learning system with a quantifiable privacy guarantee using LDP. Erlingsson *et al.* [13] described an algorithm whose privacy cost is polylogarithmic in the number of user value changes. The major drawback of LDP is its negative impact on accuracy, but efforts are being made to reduce it. Li *et al.* [24] propose a federated tree boosting framework based on order-preserving desensitization with distance-based LDP (dLDP). The distance metric introduced by dLDP reduces the amount of noise needed, as it provides a reasonable way to ignore the extreme cases where two samples are too far away, i.e, too easy to be distinguished by an adversary. However, existing dLDP mechanisms still rely on external randomness source and complete analytic forms of queries.

Empirical Differential Privacy In 2009, Duan [11] introduced the notion of privacy without noise, which estimates the corresponding (ϵ, δ) from the inherent randomness in the dataset and the query. Although this approach needs strong assumptions about data distribution, i.e., independent identically distributed data, it highlights that non-uniform internal noise can also achieve reasonable

DP. Moreover, Duan's research also inspired Burchard *et al.* to formalize empirical differential privacy (EDP) in 2019 [8]. EDP introduces a method to empirically estimate the privacy of a given query $f()$ applied to a random database S, which iterates over all adjacent data bases S_i for each sample x_i to estimate the centralized privacy risk. However, a large gap still exists between EDP and the evaluation of privacy in VFL, especially the privacy provided by each individual bottom model. First, formed as centralized DP, the estimation in EDP needs complete knowledge about every feature of each sample, contradicting with the settings of VFL where each bottom model owner knows only its own features. Second, a large number of the numerical integration of the estimated probability density function are necessary, leading to high computational cost and accumulated error even if the data set is of moderate size.

Other Existing Defenses Besides HE and multi-party computation [20], differential Privacy (DP) [12] has also been applied to the training process: Differentially Private-Stochastic Gradient Descent (**DP-SGD**) [1]. DP-SGD has been widely used as an effective and provable privacy protection mechanism to protect training data. However, and similarly to LDP, it can have a significant impact on utility. In the context of VFL, simpler mechanisms have also been proposed that can protect privacy and improve communication costs. Particularly, Noisy Gradient (**NG**) [39] adds Gaussian noise to the backward propagation, Gradient Compression (**GC**) [18] retains only the highest gradients, and Discrete SGD (**DSGD**) [5] discretizes gradients into a small number of bins. We mainly use these non-cryptographic defenses as comparison in the experiments.

Although VAE and its variants are widely used and studied in machine learning [2, 19, 27, 35], to the best of our knowledge Scheliga *et al.* [31] first proposed the use of VAE for privacy in training in 2022 as PRECODE, an VAE-based privacy-enhancing module for existing models. Although PRECODE's privacy protection against gradient inversion attack is empirically illustrated in training, Scheliga *et al.* did not give any formal interpretation of the privacy guarantee or conduct any attack experiments for the inference phase.

3 Distance-based Empirical Local Differential Privacy

3.1 Formal Definition of dELDP

We consider a threat model where a semi-honest adversary $\mathcal{A}$ can control some participants. Due to the sample alignment in VFL[‡], it is trivial to see whether an individual is in the dataset or not. Thus, in the sense of centralized DP [12] or EDP [8], VFL cannot have any meaningful privacy. Moreover, $\mathcal{A}$ against a trained VFL model M^* has two extra advantages: (1) $\mathcal{A}$ can get the label produced by M^* if $\mathcal{A}$ participates in the inference; (2) $\mathcal{A}$ can have partial information of the sample

‡ In Step 1 in Figure 1, features of the same individual have to be aligned by the identifier. If $\mathcal{A}$ participates in training, $\mathcal{A}$ sees the (quasi-)identifier of the sample.

x. Due to (1) and (2), even if we apply LDP in VFL, $\mathcal{A}$ can trivially distinguish two samples of *different* labels with the probability close to the accuracy of M^*.

Therefore, besides the range of the output, we have to consider the partial information held by $\mathcal{A}$ (leakage, especially labels), which can be formulated as the distance between two samples. We start by recalling the distance-based LDP definition and discussing its limitation.

Definition 1 (Distance-based LDP, dLDP) *We say that a dataset* S *has* (ϵ, δ)*-distance-based local differential privacy against an adversary* $\mathcal{A}$ *with regard to statistical function* $f : \mathsf{S} \to \mathcal{R}$*, distance metric* $\mathsf{dist}()$*, distance threshold* t*, if* $\forall x, x' \in \mathsf{S}, \forall y \in \mathcal{R}:$

$$\mathsf{dist}(x, x') \leq t \Rightarrow \Pr[f(x) = y] \leq e^{t\epsilon} \Pr[f(x') = y] + \delta \tag{1}$$

Given an appropriate $\mathsf{dist}()$, if we have an input-independent $f()$ with an efficiently computable analytic form, we can derive the dLDP parameters directly.

However, in VFL, the complexities inside the parameterized function $f_\theta()$ hinders the use of dLDP for quantifying the corresponding internal randomness. The value $f_\theta(x)$ depends on both θ and x, where θ, the trained model parameters, is depend on x and other samples (features) in the training data set. As mentioned before, when $f_\theta()$ spans several linear and non-linear layers, deriving the analytic form of $f_\theta()$ or the probability function of it is intractable [19]. Moreover, samples are split into features and the bottom models are trained for different features, i.e., not only the value, the function $f_\theta()$ itself may also vary for different x, adding another layer of complexity.

Thus, we opt for an *empirical* approach to circumvent the theoretical obstacles. Intuitively, if we can efficiently estimate the probability function of the outputs, we can have a meaningful bound of the privacy parameters.

Definition 2 (Distance-based Empirical LDP, dELDP) *We say that a dataset* S *has* (ϵ, δ)*- distance-based* ***empirical*** *local differential privacy against an adversary* $\mathcal{A}$ *with regard to statistical function* $f : \mathsf{S} \to \mathcal{R}$*, distance metric* $\mathsf{dist}()$*, distance threshold* t*, and* ***estimator*** $\mathsf{esmt}()$*, if*

$$\begin{aligned} &\forall x, x' \in \mathsf{S}, \forall y \in \mathcal{R} : \mathsf{dist}(x, x') \leq t \Rightarrow \\ &\qquad \mathsf{esmt}(\Pr[f(x) = y]) \leq e^{t\epsilon} \cdot \mathsf{esmt}(\Pr[f(x') = y]) + \delta \end{aligned} \tag{2}$$

Note that for the same $\mathsf{dist}()$ and t, if the estimation is perfect, i.e., $\forall x$: $\mathsf{esmt}(\Pr[f(x) = y]) = (\Pr[f(x) = y)$, then (ϵ, δ)-dELDP implies (ϵ, δ)-dLDP.

In the application of dELDP in practice, besides well-defined $\mathsf{dist}()$ and threshold, the estimation algorithm provides another degree of freedom and it can be adapted to individual VFL task for improved efficiency. Next we elaborate how dELDP can be applied to Gaussian-type internal randomness.

3.2 Application: dELDP for Trained VAEs in VFL

The posterior sampling process inside VAE has a Gaussian form, so we can estimate the *sensitivity* of the function and the *variance* instead. To see how this can proceed, first recall the Gaussian Mechanism in LDP and the posterior sampling in VAE.

Definition 3 (Gaussian Mechanism for LDP [32]) *If a function $f(x)$ of an input $x \in \mathsf{S}$ is to be released, the Gaussian release mechanism is defined as*

$$G(x) := f(x) + \mathcal{N}(0, \sigma^2 I) = f(x) + \sigma\mathcal{N}(0, I). \tag{3}$$

Theorem 1 ((ϵ, δ) of Classical Gaussian Mechanism [12, 32]) *If the sensitivity of the function is bounded by Δ_f, i.e., $\forall x, x' \in \mathsf{S}$, $||f(x) - f(x')||_2 \leq \Delta_f$, then for any $\delta \in [0, 1]$, the Gaussian mechanism $G()$ satisfies (ϵ, δ)-LDP, where*

$$\epsilon = \frac{\Delta_f}{\sigma}\sqrt{2\log\frac{1.25}{\delta}}. \tag{4}$$

Definition 4 (Posterior Sampling in VAE [19]) *Let $x \in \mathsf{S}$ be the input to VAE. The posterior sampling of $z(x)$ is defined as*

$$z(x) := \mu(x) + \Sigma(x)^{\frac{1}{2}}\mathcal{N}(0, I), \tag{5}$$

where $\mu(x)$ is the mean function and $\Sigma(x)$ the covariance matrix dependent on x learned by the VAE.

The key difference between (5) and the original Gaussian mechanism (3) is that the perturbation in (5) is a function of the input value x. Following the approach taken by [11] for non-uniform perturbation, we can generalize the Gaussian mechanism as follows.

Definition 5 (Extended Gaussian Mechanism, EG) *Let $\mathbb{R}$ be the set of real numbers. For $f : \mathsf{S} \to \mathcal{R} \subset \mathbb{R}^m$, if a function $f(x)$ of an input $x \in \mathsf{S}$ is to be released, the extended Gaussian release mechanism is defined as*

$$EG(x) := f(x) + M(x)\mathcal{N}(0, I), \tag{6}$$

where $M : \mathsf{S} \to \mathbb{R}^{m\times m}$ maps x to a diagonal matrix with non-negative entries.

By taking $f() := \mu()$, VAE posterior sampling is an instance of EG. As $\Sigma(x)^{\frac{1}{2}}$ is a diagonal matrix in VAE [19], we can collect the diagonal elements into a vector $V(x)$. The $EG(x)$ implemented by VAE can then be re-written as

$$EG(x) = f(x) + V(x) \odot \mathcal{N}(0, I). \tag{7}$$

We keep this $V(x)$ notation and omit the element-wise product symbol $\odot$ henceforth. What remains to be quantified for dLDP for VAE is an appropriate distance function and sensitivity. As we consider samples with the same label, we define

$$\mathsf{dist}(x, x') = \begin{cases} 1, \mathsf{Label}(x) = \mathsf{Label}(x') \\ \infty, otherwise \end{cases}. \tag{8}$$

Definition 6 (Local Sensitivity, Distance-based) *For* $f : \mathsf{S} \to \mathcal{R} \subset \mathbb{R}$ *and* $\forall x, x' \in \mathsf{S}$ *with* $\mathsf{Label}(x) = \mathsf{Label}(x') = \ell$*, the local sensitivity* $\mathsf{Sens}_{f,\ell}$ *of* $f()$ *with respect to* ℓ *is defined as*

$$\mathsf{Sens}_{f,\ell} = \max_{x,x'}(||f(x) - f(x')||_2). \tag{9}$$

Using (4), (7), (8) and (9), we arrive at the bound of dLDP parameters of VAE in Theorem 2. Let $\mathsf{Label(S)}$ be the set of labels of samples in S.

Theorem 2 (dLDP for VAE of each bottom model) *Given dataset* S *and query* $\mu : \mathsf{S} \to \mathcal{R} \subset \mathbb{R}^m$*, if* $\forall \ell \in \mathsf{Label(S)}, x, x' \in \mathsf{S}$, $\mathsf{Label}(x) = \mathsf{Label}(x') = \ell$ $\exists \varDelta_\ell, \sigma_\ell : \Big(\mathsf{Sens}_{\mu,\ell} \leq \varDelta_\ell \wedge \min(||V(x)||_2, ||V(x')||_2) \geq \sigma_\ell\Big)$*, then there exists* $\varDelta_\mu$ *and* σ_μ *for VAE posterior sampling for fixed* (μ, V) *that satisfy* (ϵ, δ)*-*dLDP *in the sense of Definition 1 for* $t = 1$*,* $\delta \in [0, 1]$*, and*

$$\epsilon = \frac{\varDelta_\mu}{\sigma_\mu}\sqrt{2\log\frac{1.25}{\delta}}. \tag{10}$$

Proof. (Sketch) $\varDelta_\ell$ is the upper bound of the sensitivity, and σ_ℓ is the lower bound of the (standard) variance. The intuition is that if for a label ℓ, the minimum noise is no less than $\mathcal{N}(0, \sigma_\ell^2 I)$, then the privacy guarantee is at least as good as provided by $\mathcal{N}(0, \sigma_\ell^2 I)$ with the corresponding $(\epsilon_\ell, \delta_\ell)$. And $\varDelta_\mu/\sigma_\mu$ is the worst case, i.e., the maximum among all ℓ's. The complete proof is in Appendix A. □

The last gap between dLDP and dELDP is an efficient estimation. If the size of the dataset and the number of parameters of a VAE can be bounded by a polynomial of the input length, the sensitivity and the variance can be efficiently estimated. Thus, we have the following lemma for dELDP of VAE.

Lemma 1 (dELDP for VAE). *The posterior sampling in VAE has* (ϵ, δ)*-*dELDP *in the sense of Definition 2 with*

$$\epsilon = \mathsf{esmt}\left(\frac{\varDelta_\mu}{\sigma_\mu}\right)\sqrt{2\log\frac{1.25}{\delta}}, \tag{11}$$

if $\mathsf{esmt}()$ *can be computed in polynomial time.*

Algorithm 1 plays the central role in $\mathsf{esmt}()$ which bounds the sensitivity and the variance in the trained VAE with respect to the given partial dataset. Let the number of labels be a constant, and the number of data items be n_x. The time complexity of Algorithm 1 is $\mathcal{O}(n_x^2)$. The computation includes only inner-product and comparison. Once $\{(\varDelta_\ell, \sigma_\ell)\}$ is collected, the term $\mathsf{esmt}\left(\frac{\varDelta_\mu}{\sigma_\mu}\right)$ can be computed as $\mathsf{max}\left(\frac{\varDelta_\ell}{\sigma_\ell}\right)$. The empirical results will be presented in Table 3 in Section 5.

Algorithm 1 dELDP for VAE

Input: P_i's partial data S_i and $\mathsf{Label}(\mathsf{S})$;
Output: $\{(\Delta_\ell/\sigma_\ell)\}$, the maximum sensitivity over the minimum variance pairs of each label $\ell \in \mathsf{Label}(\mathsf{S})$ with respect to S_i.
Algorithm Starts:
for each label $\ell \in \mathsf{Label}(\mathsf{S})$ **do**
    $\Delta_\ell = 0, \sigma_\ell = 0$
    for each pair (x, x') in S_i **do**
        if $\mathsf{Label}(x) = \mathsf{Label}(x') = \ell$ **then**
            $\Delta = ||\mu_x - \mu_{x'}||_2$, $\sigma = \min(||V(x)||_2, ||V(x')||_2)$;
            $\Delta_\ell = \max(\Delta_\ell, \Delta)$, $\sigma_\ell = \min(\sigma_\ell, \sigma)$;
        end if
    end for
end for
Output: $\{(\Delta_\ell/\sigma_\ell)\}$ for S_i

4 Implementation

Rather than testing existing models that contains VAE-like structures, we opt for a constructive approach: we add VAE to VFL that do not contain any other sources of randomness, so the effect of internal randomness can be observed without interference. In principle, we start from efficient baseline VFL bottom and top models, and we insert and appropriate VAE between them after adapting the hyper-parameters. More specifically, all the VAEs share the output dimensions of the corresponding bottom model, and each VAE has the same shape of input and output (See Figure 2). Further implementation for training details can be found in Algorithm 2. The inference is identical to the forward pass except that the loss function is not computed.

We denote the VFL pipeline extended with VAE as SAIR, as it SAves the cost with Internal Randomness. On the other hand, SAIR can be further extended with other orthogonal privacy-enhancing mechanisms. For example, when each individual embedding must be protected against strong adversaries, secure aggregation with malicious security [23] can be applied. If the secrecy of top model parameters is critical, using multi-party computation [20] for the top model is a good option. Analyzing the overall privacy of other extended pipelines is intriguing but not in the scope of this paper. Let $\{\theta_{\mathcal{C}_i}\}$ and $\theta_{\mathcal{S}}$ be the parameters of model $\mathsf{M} = (\mathsf{M_B}, \mathsf{M_T})$. The update strategy is as follows.

$$\theta_{\mathcal{C}_i} = \theta_{\mathcal{C}_i} - \eta \frac{\partial \mathcal{L}(out_T, \mathsf{Label})}{\partial \theta_{\mathcal{C}_i}} \tag{12}$$

$$\theta_{\mathcal{S}} = \theta_{\mathcal{S}} - \eta \frac{\partial \mathcal{L}(out_T, \mathsf{Label})}{\partial \theta_{\mathcal{S}}} \tag{13}$$

Algorithm 2 Training Process in SAIR

Input and Parameters:
Top model owner $\mathcal{S}$ and N bottom model owners, a dataset S, and labels $\mathsf{Label(S)}$;
Bottom model, owner P_i, has partial data S_i, $\forall i \in [N]$;
(Pre-trained) bottom model $\mathsf{M_B}$ with VAE, top model $\mathsf{M_T}$;
Top and bottom model parameters θ_S, $\{\theta_{C_i}, \forall i \in [N]\}$, resp.;
Loss function $\mathcal{L}$ of final outputs, loss functions $\{\mathcal{L}_{(\mu_i,\sigma_i)}\}$ of bottom models, parameters $\{\mu_i\}$ and $\{\sigma_i\}$ of VAEs;
Learning rate η, total training epochs $\mathcal{E}$.
Algorithm Starts:
for $e = 1, 2, ..., \mathcal{E}$ **do**
  // Training for bottom model
  for each bottom model owner P_i **do**
    // 1. Forward propagation for bottom models
    // Train $\mathsf{M_B}$ in parallel;
    $out_i^e = \mathsf{M_B}\ (\mathsf{S}_i)$;
    Compute $\mathcal{L}_{(\mu_i,\sigma_i)}$;
    Send out_i^e to $\mathcal{S}$; //All out_i^e have the same shape.
    // 2. Back propagation and stochastic gradient descent for bottom models
    Compute $\partial\mathcal{L}_(\mu_i, \sigma_i)$, update μ_i and σ_i;
    // Until the server finishes backward propagation
    Receive $\frac{\partial\mathcal{L}(out_T,\mathsf{Label})}{\partial out_i^e}$;
    Compute $\frac{\partial\mathcal{L}(out_T,\mathsf{Label})}{\partial\theta_{\mathcal{C}_i}} = \frac{\partial\mathcal{L}(out_T,\mathsf{Label})}{\partial out_i^e} \times \frac{\partial out_i^e}{\partial\theta_{\mathcal{C}_i}}$;
    Update $\theta_{\mathcal{C}_i} = \theta_{\mathcal{C}_i} - \eta\frac{\partial\mathcal{L}(out_T,\mathsf{Label})}{\partial\theta_{\mathcal{C}_i}}$;
  end for
  // Training for top model
  // 1. Forward propagation for top model
  $out_T = \mathcal{M}_T(out_i^e)$;
  Compute $\mathcal{L}(out_T, \mathsf{Label(S)})$;
  // 2. Backward propagation for top model;
  $\theta_\mathcal{S} = \theta_\mathcal{S} - \eta\frac{\partial\mathcal{L}(out_T,\mathsf{Label(S)})}{\partial\theta_\mathcal{S}}$;
  Compute and send $\frac{\partial\mathcal{L}(out_T,\mathsf{Label})}{\partial out_i^e}$ to all bottom model owners;
end for

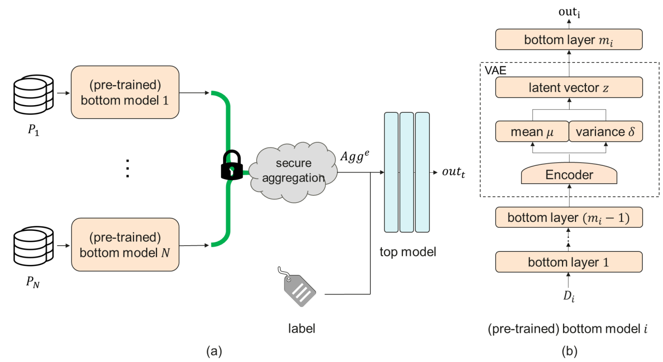

Fig. 2: Overview of VAE-enhanced VFL: (a) data flow; (b) VAE in bottom model. Secure aggregation [7] against semi-honest adversaries is used.

5 Experiments

5.1 Setup

Datasets. We use six real-world multiclass classification datasets for evaluation: Sensorless Drive Diagnosis (**SDD**) [4], **Criteo** [22], **MNIST** [9], **MedicalMNIST** [36], Look and Listen (**LL**, a multimodal dataset with image-audio modalities) [6], and **AG's News** Topic Classification Dataset [38]. The statistics of the six datasets are shown in Table 1. All experiments were run on a Tesla™P100 GPU with 16GB RAM.

Baseline Models. We build up a neural network with four fully connected layers for **SDD**, **Criteo**, **MNIST** and **MedicalMNIST** with different input and output. We divide the first two layers for the two bottom models and the last two layers for the top model.

Table 1: Bottom and Top Model Parameters. Different clients hold different ratios of features, for example, client 0 and client 1 have 24 features in **SDD**.

Dataset	**#Samples**	**#Labels**	**#Params. Bottom Model Feature Distribution**	**#Params. Top Model**
SDD	58.5K	11	[**18.5K, 18.5K**] (24, 24)	**6K**
Criteo	44841K	2	[**17.5K, 18.1K**] (13, 26)	**5K**
MNIST	70K	10	[**72.4K, 72.4K**] (14, 14)	**10.8K**
MedicalMNIST	59K	6	[**315K, 315K**] (16, 16)	**10.8K**
LL	17.2K	9	[**294M, 733K**] (image, audio)	**1.7K**
AG's News	128K	4	[**1.5M, 1.5M**] ('title', 'description')	**32.3K**

Performance Experiments. We measure the execution time of different settings of SAIR for training, i.e., baseline (without any protection), using only the Variational Autoencoder (VAE), and using a pre-trained VAE (see Table 2).

dELDP Experiments. We quantify dELDP of VAE using Algorithm 1. The algorithm is executed over each label and each bottom model trained on **SDD**, **MedicalMNIST**, **LL** and **AG's News**. The primary results from the experiments are the maximum values $\frac{\Delta_\ell}{\sigma_\ell}$ for each label-model combination.

Attack Experiments. We evaluate the VAE extended VFL system against the following state-of-the-art attacks applicable during model inference.

- **Feature reconstruction attack (FR).** A FR needs access to the trained model. We implemented FR under the *white-box* setting [17], which provides a stronger adversary and an upper bound of privacy loss. In this attack, the adversary $\mathcal{A}$ uses backpropagation to update a random input, as to minimize the distance between the true confidence scores and the output for this input. $\mathcal{A}$ collects the true confidence scores of the trained model during inference, meanwhile forming pairs of partial inputs and respective outputs.
- **Label Inference attack (LI)** As LI, we implement the Passive Label Inference Attack through Model Completion [14] without access to the complete trained model. In this LI, adversary $\mathcal{A}$ tries to infer the labels of new samples by using its own bottom model $\mathsf{M}_\mathsf{B}^\mathcal{A}$, and complementing that model with an *inference head*, which is randomly initialized additional layers that complete the local model. $\mathcal{A}$ trains this head by freezing $\mathsf{M}_\mathsf{B}^\mathcal{A}$ and using samples containing only $\mathcal{A}$'s training data (i.e., part of the features) and the respective labels. We make a strong assumption that the inference head has the same architecture as the top model, and $\mathcal{A}$'s data is the validation dataset. The testing data is then used to measure the performance.
- **Model stealing (MS).** MS attackers train shadow models and need access to the trained model. We implemented the attack in [28]. In addition, the adversary $\mathcal{A}$ has some input data with all features, which is referred to as the *background information*. $\mathcal{A}$ trains a new model M' by querying the victim model M^* and uses the difference between both outputs to update M'. We assume a strong $\mathcal{A}$, whose M' has the same architecture as M^* but pre-trained on a different dataset.
  - For stealing M^* trained with **MNIST**, we use the **MedicalMNIST** dataset to pre-train the adversarial model M';
  - For stealing M^* trained with LL, we use the **CIFAR10** dataset to pre-train M'.

  Following the approach taken by [28], we use three different background information for MS:
  1. the full training set S, i.e., $\mathcal{A}$'s background information covers all the (private) training data S. It should be noted that if $\mathcal{A}$ has the full dataset, he can train the model from scratch. Therefore this is an unrealistic scenario to provide an upper bound on the attack performance;

2. the validation set S_V, i.e., $\mathcal{A}$'s background information has *no* intersection with S;
3. a randomly sampled subset from the S of the size $|\mathsf{S}_V|$, i.e., $\mathcal{A}$'s background information intersects with part of S.

We refer to [14] and consider four privacy-preserving methods as our experimental comparisons: Noisy Gradient **NG** [39], Gradient Compression **GC** [18], Differentially Private - Stochastic Gradient Descent **DP-SGD** [1] and Discrete Stochastic Gradient Descent **DSGD** [5].

NG is a common strategy with adding noise to the gradients [39]. We refer to [14] and consider four NG ratios for comparisons, namely 10^{-4}, 10^{-3} and 10^{-2}.**GC** is a method for privacy protection and communication efficiency [18]. We consider 75%, 50%, 25% and 10% as compression rates. **DP-SGD** is widely used in privacy-preserving deep learning [1]. We choose ϵ is 0.1, 1 and 10. **DSGD** protects the signs of gradients. Thus, it benefits not only the privacy protection but also contributes to the communication efficiency [5]. We choose **DSGD** bins are 6, 12, 18 and 24, which refers to [14].

5.2 Result Summary

Performance. To analyze the performance, we measure the accuracy change over epochs in Figure 3 and total execution time in Table 2.

Overall, the results confirm that using SAIR with pre-trained model to replace the insecure baseline model has minimum impact on the training and inference cost. Figure 3 shows that SAIR with VAE does not affect convergence in training. Moreover, pre-trained models accelerate the speed of convergence. Table 2 shows that the basic SAIR consumes a maximum up to 1.43 times than baseline. The acceleration from a pre-trained model is also significant. The time cost is c.a. 1.04 to 1.40 of that of the baseline.

Table 2: Time consumption (in minutes) till convergence in each setting. The time used by Baseline is taken as 1.00x.

Settings / Datasets	SDD	Criteo	MNIST	MedicalMNIST	LL	AG's News
Baseline (not secure)	44.04 (1.00x)	251.13 (1.00x)	31.31 (1.00x)	39.00 (1.00x)	109.17 (1.00x)	296.71 (1.00x)
Basic SAIR	57.68 (1.30x)	258.21(1.03x)	44.96 (1.43x)	47.85 (1.23x)	141.38 (1.29x)	370.57 (1.24x)
SAIR **(Pre-trained)**	50.20 (**1.14x**)	266.20 (**1.06x**)	43.41 (**1.40x**)	46.78 (**1.21x**)	118.89 (**1.09x**)	309.08 (**1.04x**)

The dELDP of VAE. Table 3 shows the results of Algorithm 1 for $\frac{\Delta_\ell}{\sigma_\ell}$ in Section 4.1. The ratio $\frac{\Delta_\ell}{\sigma_\ell}$ lies between 0.1 and 2.1. **SDD** gives us the maximum ratio at client 0 on label 6. Meanwhile, **MedicalMNIST** gives us the minimum at client 1 on label 1. The ratios are more evenly distributed around 0.5 with **LL**. Besides, **AG's News** holds the ratios above 1.1 besides label 4. By setting $\delta = 2^{-32}$, we can obtain $\epsilon \approx 2.59$ to 10.23 from the table.

In general, we estimate $\epsilon \approx 6.4$ with a $\delta = 2^{-32}$ on average.

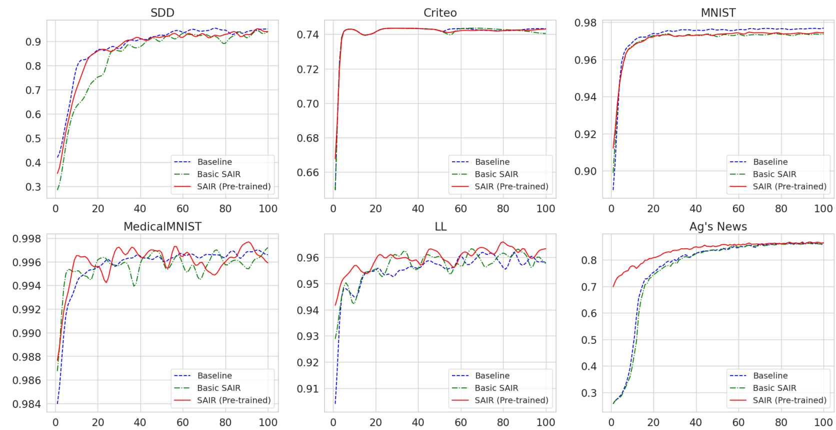

Fig. 3: Accuracy change in training over different datasets. Models with pre-trained VAE and correlated randomness converge faster than baseline and those without pre-trained models. All experiments run 100 epochs.

Table 3: dELDP results, the maximum $\frac{\Delta_\ell}{\sigma_\ell}$ values for each label-model combination. The largest value on each client is marked in bold.

Dataset	Labels & Clients	Ratio										
SDD	**Label**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
	Client 0	0.5381	0.5611	0.2431	0.4871	0.4032	**2.1041**	0.2155	0.5046	0.7639	0.4732	0.4926
	Client 1	0.6210	0.5126	0.3827	0.5971	0.6307	**1.9711**	0.4227	0.7162	0.8985	0.4009	0.4071
MedicalMNIST	**Label**	**1**	**2**	**3**	**4**	**5**	**6**					
	Client 0	0.2430	0.3360	0.8323	0.1992	1.1190	**1.8725**					
	Client 1	0.1092	0.2079	**1.2161**	0.1474	0.8595	0.3430					
LL	**Label**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		
	Client 0	0.4781	0.7276	0.5979	**0.7690**	0.7464	0.4585	0.7487	0.7345	0.6337		
	Client 1	0.4869	0.4273	0.4070	**0.5339**	0.4711	0.4410	0.4846	0.4188	0.4644		
AG's News	**Label**	**1**	**2**	**3**	**4**							
	Client 0	1.3804	**1.7998**	1.6909	0.4732							
	Client 1	1.1874	1.3442	**1.3543**	0.4009							

Resilience against Attacks. As aforementioned, we executed three attacks: feature reconstruction (FR), label inference (LI) in Table 4, and model stealing (MS) in Table 5. For comparison, we consider both the model utility and the resilience, as a model with low accuracy is usually more robust against attacks but not useful.

In Table 4, we compare SAIR with models protected by **NG**, **GC**, **DSGD**, and **DP-SGD** against FR and LI. When we use NG with a noise scale of 0.01 or 0.1, it results in low convergence and further affects the attack success rate. **DP-SGD** has a more powerful defense than **GC**, **DSGD**, and **NG**, as clearly observed from the label inference accuracy, but it also incurs significant utility loss. When ϵ equals 0.1, 1, and 10, the baseline accuracy is 91.70%, 95.57% and 96.80%, respectively.

As ϵ and accuracy increase, the defense becomes less effective. When the adversary has 50% feature capacity in a label inference attack, the accuracy *distance* is 34.20%, 22.76% and 16.42%, respectively. In comparison with using no defense and $\epsilon = 10$, SAIR has higher accuracy while having the efficiency of **DP-SGD** with $\epsilon = 10$. In addition, SAIR achieves notable efficiency, even when the adversary holds a large proportion of features. When the adversary has 75% features and no protection, the label extraction accuracy is 97.02%. Meanwhile, SAIR has a label extraction accuracy of only 87.68%. The last row of Table 4 also shows that the internal randomness in VAE can be empirically combined with **DP-SGD** and achieve better resilience against attacks.

When facing MS, SAIR in general outperforms **DP-SGD** in both the utility and protection metrics. In comparison with the baseline, SAIR has an accuracy loss below 0.5%, while **DP-SGD** drops up to 16% in the in **MedicalMNIST** (83.06% versus the baseline 99.9%). With regard to the attack, SAIR also outperforms **DP-SGD** in provided protection when using the validation data or the training subsamples, with a minimum observed attacker accuracy of 11.52% in the **MNIST** dataset. We use the full training data as the strongest attacker reference, which shows two special cases in **MedicalMNIST**. When the task accuracy is 83.06%, the adversary is not able to achieve better result (82.36%). However, when the task accuracy is very good (99.96%) because of overfitting, the adversary is then powerful to achieve a good result (97.63%).

Table 4: **MNIST**: Adversary performance for the feature and label extraction attacks against the different defenses, and considering an adversary holding different ratios of features. The adversaries hold 25%, 50% and 75% of the total feature space. Note that for feature extraction we use the Mean Squared Error, whereas for the label extraction, we use the Accuracy. Note that we use the Mean Square Error (MSE) for feature extraction, whereas for the label extraction, we use the Accuracy (Acc).

Defense Approach	Parameter	Parameter Set Value	Model Accuracy (50%)	FR; MSE (Adv. is active party)			LI; Acc (Adv. is passive party)		
				25%	50%	75%	25%	50%	75%
	No Defense		97.86%	2.46	2.32	1.37	35.05%	88.30%	97.02%
NG	Noise Scale	1e-4	97.34%	2.32	2.12	1.40	10.20%	82.74%	95.56%
		1e-3	96.88%	2.34	2.13	1.57	9.11%	82.48%	95.52%
		1e-2	94.17%	2.71	3.07	2.13	10.14%	76.43%	82.70%
GC	Compression Rate	75%	97.85%	2.33	2.34	1.39	9.54%	86.71%	95.01%
		50%	97.87%	2.33	2.32	1.39	10.74%	84.07%	95.50%
		25%	97.85%	2.35	2.31	1.38	11.49%	86.65%	95.92%
		10%	97.82%	2.35	2.32	1.41	9.82%	85.64%	95.56%
DSGD	Bin	24	97.42%	2.34	2.34	1.41	9.86%	85.14%	95.12%
		18	97.32%	2.36	2.38	1.50	9.62%	85.59%	92.90%
		12	97.42%	2.37	2.44	1.38	11.26%	79.43%	95.09%
		6	97.43%	2.61	2.29	1.45	6.74%	80.72%	94.12%
DP-SGD	ϵ	0.1	91.70%	2.28	2.18	1.37	9.31%	57.50%	71.85%
		1	95.57%	2.38	2.24	1.35	12.62%	72.81%	79.15%
		10	96.80%	2.40	2.30	1.44	10.07%	80.38%	87.03%
	Ours		97.68%	2.34	2.15	1.36	9.67%	81.66%	87.68%
	Ours & DP-SGD($\epsilon = 10$)		96.94%	2.42	2.28	1.43	9.63%	77.18%	86.15%

Table 5: Model Stealing Attacks

Dataset	Protection	Task Accuracy	Attack Accuracy		
			Full Training Data	Validation Data	Training Subsamples
MNIST	Baseline	97.68%	77.42%	33.44%	33.61%
	DP-SGD(ϵ=10)	93.83%	76.95%	32.83%	32.82%
	Ours	97.53%	67.80%	11.52%	11.54%
MedicalMNIST	Baseline	99.90%	98.93%	97.06%	96.73%
	DP-SGD (ϵ=0.1)	83.06%	82.36%	80.15%	80.60%
	Ours	99.96%	97.63%	79.96%	80.43%
LL	Baseline	95.63%	92.81%	90.14%	90.43%
	DP-SGD (ϵ=0.1)	94.72%	91.51%	89.98%	89.54%
	Our Solution	95.16%	88.28%	86.55%	86.14%

6 Conlusion and Future Work

We have developed a new differential privacy estimate dELDP that formally captures the privacy guarantee of internal randomness inside VFL algorithms. The experiments demonstrate that dELDP is also practical and can lead to efficient defenses against various attacks in VFL inference.

We still wonder whether it is possible to compose the parameters of both worlds, i.e., form a generic bound given both the LDP and dELDP mechanisms robustly in theory. To find more precise bound of the estimation and to develop concrete algorithms for other types of internal randomness, e.g., components within Bayesian networks [33], can be meaningful future work.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 308–318 (2016)
2. Bai, J., Wang, W., Gomes, C.P.: Contrastively disentangled sequential variational autoencoder. Advances in Neural Information Processing Systems **34**, 10105–10118 (2021)
3. Barker, E.: Recommendation for key management. NIST special publication 800-57 part 1 revision 5 (2020)
4. Bator, M.: Dataset for Sensorless Drive Diagnosis. UCI Machine Learning Repository (2015), DOI: https://doi.org/10.24432/C5VP5F
5. Bernstein, J., Wang, Y.X., Azizzadenesheli, K., Anandkumar, A.: signsgd: Compressed optimisation for non-convex problems. In: International Conference on Machine Learning. pp. 560–569. PMLR (2018)
6. Bird, J.J., Faria, D.R., Premebida, C., Ekárt, A., Vogiatzis, G.: Look and listen: A multi-modality late fusion approach to scene classification for autonomous machines. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 10380–10385. IEEE (2020)
7. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)

8. Burchard, P., Daoud, A., Dotterrer, D.: Empirical differential privacy. arXiv preprint arXiv:1910.12820 (2019)
9. Cohen, G., Afshar, S., Tapson, J., Van Schaik, A.: Emnist: Extending mnist to handwritten letters. In: 2017 international joint conference on neural networks (IJCNN). pp. 2921–2926. IEEE (2017)
10. Dai, Y., Qian, Y., Lu, F., Wang, B., Gu, Z., Wang, W., Wan, J., Zhang, Y.: Improving adversarial robustness of medical imaging systems via adding global attention noise. Computers in Biology and Medicine **164**, 107251 (2023)
11. Duan, Y.: Privacy without noise. In: Proceedings of the 18th ACM conference on Information and knowledge management. pp. 1517–1520 (2009)
12. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)
13. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., Thakurta, A.: Amplification by shuffling: From local to central differential privacy via anonymity. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 2468–2479. SIAM (2019)
14. Fu, C., Zhang, X., Ji, S., Chen, J., Wu, J., Guo, S., Zhou, J., Liu, A.X., Wang, T.: Label inference attacks against vertical federated learning. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 1397–1414 (2022)
15. Geng, J., Mou, Y., Li, F., Li, Q., Beyan, O., Decker, S., Rong, C.: Towards general deep leakage in federated learning. arXiv preprint arXiv:2110.09074 (2021)
16. Grining, K., Klonowski, M.: Towards extending noiseless privacy: Dependent data and more practical approach. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 546–560 (2017)
17. Jiang, X., Zhou, X., Grossklags, J.: Comprehensive analysis of privacy leakage in vertical federated learning during prediction. Proc. Priv. Enhancing Technol. **2022**(2), 263–281 (2022)
18. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. Foundations and Trends in Machine Learning (2021)
19. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
20. Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., van der Maaten, L.: Crypten: Secure multi-party computation meets machine learning. Advances in Neural Information Processing Systems **34**, 4961–4973 (2021)
21. Koker, T., Mireshghallah, F., Titcombe, T., Kaissis, G.: U-noise: Learnable noise masks for interpretable image segmentation. In: 2021 IEEE International Conference on Image Processing (ICIP). pp. 394–398. IEEE (2021)
22. Labs., C.: Criteo dataset. Online (February 12 2021), .`https://labs.criteo.com/2014/02/download-kaggle-display-advertising-challenge-dataset/`
23. Li, H., Lin, H., Polychroniadou, A., Tessaro, S.: Lerna: Secure single-server aggregation via key-homomorphic masking. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 302–334. Springer (2023)
24. Li, X., Hu, Y., Liu, W., Feng, H., Peng, L., Hong, Y., Ren, K., Qin, Z.: Opboost: A vertical federated tree boosting framework based on order-preserving desensitization. Proc. VLDB Endow. **16**(2), 202–215 (oct 2022). https://doi.org/10.14778/3565816.3565823, `https://doi.org/10.14778/3565816.3565823`

25. Liu, Y., Kang, Y., Zou, T., Pu, Y., He, Y., Ye, X., Ouyang, Y., Zhang, Y.Q., Yang, Q.: Vertical federated learning. arXiv preprint arXiv:2211.12814 (2022)
26. Luo, X., Wu, Y., Xiao, X., Ooi, B.C.: Feature inference attack on model predictions in vertical federated learning. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). pp. 181–192. IEEE (2021)
27. Neumeier, M., Botsch, M., Tollkühn, A., Berberich, T.: Variational autoencoder-based vehicle trajectory prediction with an interpretable latent space. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 820–827. IEEE (2021)
28. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4954–4963 (2019)
29. Ou, W., Zeng, J., Guo, Z., Yan, W., Liu, D., Fuentes, S.: A homomorphic-encryption-based vertical federated learning scheme for rick management. Computer Science and Information Systems **17**(3), 819–834 (2020)
30. Salowey, J.A., McGrew, D., Choudhury, A.: AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288 (Aug 2008). https://doi.org/10.17487/RFC5288, `https://www.rfc-editor.org/info/rfc5288`
31. Scheliga, D., Mäder, P., Seeland, M.: Precode-a generic model extension to prevent deep gradient leakage. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1849–1858 (2022)
32. Seif, M., Tandon, R., Li, M.: Wireless federated learning with local differential privacy. In: 2020 IEEE International Symposium on Information Theory (ISIT). pp. 2604–2609. IEEE (2020)
33. Shridhar, K., Laumann, F., Liwicki, M.: A comprehensive guide to bayesian convolutional neural network with variational inference. arXiv preprint arXiv:1901.02731 (2019)
34. Truex, S., Liu, L., Chow, K.H., Gursoy, M.E., Wei, W.: Ldp-fed: Federated learning with local differential privacy. In: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking. pp. 61–66 (2020)
35. Yang, C., Wang, X., Mao, S.: Autotag: Recurrent variational autoencoder for unsupervised apnea detection with rfid tags. In: 2018 IEEE Global Communications Conference (GLOBECOM). pp. 1–7. IEEE (2018)
36. Yang, J., Shi, R., Ni, B.: Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In: IEEE 18th International Symposium on Biomedical Imaging (ISBI). pp. 191–195 (2021)
37. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
38. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. Advances in neural information processing systems **28** (2015)
39. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. Advances in neural information processing systems **32** (2019)

A Proof of Theorem 2

Here is the complete proof of the dLDP main theorem (Theorem 2).

Proof. We start from one-dimensional case and upgrade it to the high dimensional case. If the distance $< \infty$, it is 1 by definition for x, x' with $\mathsf{Label}(x) =$

$\mathsf{Label}(x')$ in (8). We start from one-dimensional case and real valued $\mu(x), V(x)$. Fix a label $\ell = \mathsf{Label}(x) = \mathsf{Label}(x')$, The sensitivity is

$$\Delta_\ell = \max_{x,x'}(|\mu(x) - \mu(x')|)$$

We then consider the absolute value of the privacy loss for x, x':

$$\left|\ln\left(\frac{e^{(-1/2\sigma_1^2)x^2}}{e^{(-1/2\sigma_2^2)(x+\Delta_\ell)^2}}\right)\right|, \tag{14}$$

where $\sigma_1 = ||V(x)||_2$ and $\sigma_2 = ||V(x')||_2$, and $V()$ is fixed.[§]

Without loss of generality, we assume $\sigma_1 \leq \sigma_2$. Then we have

$$\left|\ln\left(\frac{e^{(-1/2\sigma_1^2)x^2}}{e^{(-1/2\sigma_1^2)(x+\Delta_\ell)^2}}\right)\right| \geq \left|\ln\left(\frac{e^{(-1/2\sigma_1^2)x^2}}{e^{(-1/2\sigma_2^2)(x+\Delta_\ell)^2}}\right)\right|. \tag{15}$$

So to bound (14), we can bound the left-hand side of (15) instead, i.e., we need an ϵ_1, such that

$$\epsilon_1 \geq \left|\ln\left(\frac{e^{(-1/2\sigma_1^2)x^2}}{e^{(-1/2\sigma_1^2)(x+\Delta_\ell)^2}}\right)\right| = \left|\frac{1}{2\sigma_1^2}(2x\Delta_\ell - \Delta_\ell/2)\right|. \tag{16}$$

Thus, if we iterate over all x with $\mathsf{Label}(x) = \ell$ and let

$$\sigma_\ell := \min(\{\sigma_i\}), \sigma_i = V(x_i),$$

the bound for the privacy loss w.r.t. label ℓ is

$$\epsilon_\ell \geq \left|\ln\left(\frac{e^{(-1/2\sigma_\ell^2)x^2}}{e^{(-1/2\sigma_\ell^2)(x+\Delta_\ell)^2}}\right)\right| = \left|\frac{-1}{2\sigma_\ell^2}(2x\Delta_\ell - \Delta_\ell/2)\right| \tag{17}$$

With a similar argument in the proof of Theorem A.1 in [12], and since $|x|$ is identical to $||x||_2$ if $x \in \mathbb{R}$, then we can have the relation

$$\epsilon_\ell = \frac{\Delta_\ell}{\sigma_\ell}\sqrt{2\log\frac{1.25}{\delta}}. \tag{18}$$

By taking $\epsilon = \max(\{\epsilon_\ell\})$, we can conclude that Theorem 2 is correct for one-dimensional posterior sampling in VAE.

We use a convention $\mu_x := \mu(x)$. For the high dimension case, i.e., $\mu_x, x, V(x) \in \mathbb{R}^m$, the sensitivity is changed to

$$\Delta_\ell = \max_{x,x'}(||\mu_x - \mu_{x'}||_2).$$

[§] $V(x) \in \mathbb{R}$, so $|V(x)| = ||V(x)||_2$. The VAE parameter θ has been trained.

Similarly, we are interested in the privacy loss w.r.t. label ℓ

$$\left|\ln\left(\frac{e^{(-1/(2\sigma_1^\intercal\sigma_1))||x||_2^2}}{e^{(-1/(2\sigma_2^\intercal\sigma_2))||(x+\mu_x-\mu_{x'})||_2^2}}\right)\right|, \tag{19}$$

For $\sigma_1 = V(x)$ and $\sigma_2 = V(x')$, with $\sigma_1^\intercal\sigma_1 \leq \sigma_2^\intercal\sigma_2$, we can have

$$\left|\ln\left(\frac{e^{(-1/(2\sigma_1^\intercal\sigma_1))||x||_2^2}}{e^{(-1/(2\sigma_2^\intercal\sigma_2))||(x+\mu_x-\mu_{x'})||_2^2}}\right)\right| \geq \left|\ln\left(\frac{e^{(-1/(2\sigma_1^\intercal\sigma_1))||x||_2^2}}{e^{(-1/(2\sigma_1^\intercal\sigma_1))||(x+\mu_x-\mu_{x'})||_2^2}}\right)\right|. \tag{20}$$

If we assume that $\mu_x - \mu_{x'}$ with the maximum norm is aligned with x (which gives the biggest denominator), we are back to the one-dimensional case, i.e., for

$$\sigma_\ell = \arg\min_{\sigma_i}(\sigma_i^\intercal\sigma_i),$$

the bound ϵ_ℓ for the privacy loss must fulfill

$$\epsilon_\ell \geq \left|\ln\left(\frac{e^{(-1/(2\sigma_\ell^\intercal\sigma_\ell))||x||_2^2}}{e^{(-1/(2\sigma_\ell^\intercal\sigma_\ell))(||x||_2+\Delta_\ell)^2}}\right)\right| = \left|\frac{-1}{2\sigma_\ell^\intercal\sigma_\ell}(2||x||_2\Delta_\ell - \Delta_\ell/2)\right|. \tag{21}$$

So we have (22) in high dimension. Theorem 2 holds.¶

$$\epsilon_\ell = \frac{\Delta_\ell}{\sqrt{\sigma_\ell^\intercal\sigma_\ell}}\sqrt{2\log\frac{1.25}{\delta}}. \tag{22}$$

□

¶ We abuse the notation by letting $\sigma_\ell = \sqrt{\sigma_\ell^\intercal\sigma_\ell}$ so we have the form in the theorem.