VFLGAN: Vertical Federated Learning-based Generative Adversarial Network for Vertically Partitioned Data Publication

Xun Yuan*
National University of Singapore
Singapore, Singapore
e0919068@u.nus.edu

Yang Yang
National University of Singapore
Singapore, Singapore
y.yang@u.nus.edu

Prosanta Gope*
University of Sheffield
Sheffield, United Kingdom
p.gope@sheffield.ac.uk

Aryan Pasikhani
University of Sheffield
Sheffield, United Kingdom
aryan.pasikhani@sheffield.ac.uk

Biplab Sikdar
National University of Singapore
Singapore, Singapore
bsikdar@nus.edu.sg

ABSTRACT

In the current artificial intelligence (AI) era, the scale and quality of the dataset play a crucial role in training a high-quality AI model. However, good data is not a free lunch and is always hard to access due to privacy regulations like the General Data Protection Regulation (GDPR). A potential solution is to release a synthetic dataset with a similar distribution to that of the private dataset. Nevertheless, in some scenarios, it has been found that the attributes needed to train an AI model belong to different parties, and they cannot share the raw data for synthetic data publication due to privacy regulations. In PETS 2023, Xue et al. [29] proposed the first generative adversary network-based model, VertiGAN, for vertically partitioned data publication. However, after thoroughly investigating, we found that VertiGAN is less effective in preserving the correlation among the attributes of different parties. This article proposes a Vertical Federated Learning-based Generative Adversarial Network, VFLGAN, for vertically partitioned data publication to address the above issues. Our experimental results show that compared with VertiGAN, VFLGAN significantly improves the quality of synthetic data. Taking the MNIST dataset as an example, the quality of the synthetic dataset generated by VFLGAN is **3.2** times better than that generated by VertiGAN w.r.t. the Fréchet Distance. We also designed a more efficient and effective Gaussian mechanism for the proposed VFLGAN to provide the synthetic dataset with a differential privacy guarantee. On the other hand, differential privacy only gives the upper bound of the worst-case privacy guarantee. This article also proposes a practical auditing scheme that applies membership inference attacks to estimate privacy leakage through the synthetic dataset.

KEYWORDS

Generative adversarial network, Federated learning, Differential privacy, Privacy-preserving data publication

*Corresponding authors

Proceedings on Privacy Enhancing Technologies 2024(4), 840–858

https://doi.org/10.56553/popets-2024-0144

1 INTRODUCTION

In the realm of deep learning (DL), the efficacy of DL models is intimately tied to the quality and scale of the data they are trained on. For instance, the advancements in image perception models can be largely attributed to comprehensive datasets like ImageNet [12]. Similarly, as demonstrated in [51], the state-of-the-art language understanding methods thrive on expansive textual datasets like [10]. Furthermore, the success of modern recommendation systems, highlighted in [36], hinges on rich datasets like Netflix ratings [6]. When these datasets are effectively leveraged with deep learning, the possibilities are boundless, enabling organizations and governments to devise strategies with unprecedented precision and foresight. However, a notable challenge in this realm is the nature of data collection. Often, data is 'vertically partitioned', meaning different pieces of customer information are scattered across multiple entities. For example, while a bank may hold a client's financial history, their health records might be with a hospital or insurance firm. As per studies like [29, 50], integrating such dispersed attributes can provide a holistic view of customers, thus significantly enhancing decision-making processes.

Despite the evident benefits of integrating dispersed data attributes, practical implementation is often hampered due to privacy concerns. Moreover, stringent data protection regulations like GDPR [53] further curb the sharing of customer data between entities. One potential avenue to navigate these challenges is the publication of synthetic data that mirror the distribution of private data without disclosing any actual private information. However, this solution is not without its vulnerabilities. Adversaries have devised methods that leverage synthetic datasets to glean insights into the corresponding private datasets. Cases in point are the Membership Inference (MI) attacks and attribute inference techniques proposed in [47]. To counteract such vulnerabilities, Differential Privacy (DP) [14] offers a promising strategy for privacy protection. By infusing DP principles into synthetic data publication, one can provide these synthetic datasets with a robust privacy assurance, fortifying them against threats like MI attacks.

Building upon the foundational discussions on data privacy and the challenges of vertically partitioned data publication, this paper addresses the limitations of existing DP methods in managing vertically partitioned data. Notably, conventional DP solutions like DistDiffGen [41], and DPLT [50] are tailored for specific kinds of datasets. Generative Adversarial Networks (GANs) [19], recognized

for their ability to replicate original data distributions, provide a novel approach for universal kinds of datasets. Besides, Federated Learning (FL) can help to comply with GDPR's data localization requirements. Through the integration of GANs with Vertical Federated Learning (VFL) [33], our proposed VFLGAN model surpasses VertiGAN [29] in achieving enhanced attribute correlation. Furthermore, we extend this innovation with DP-VFLGAN, incorporating an optimized Gaussian mechanism tailored to the VFL context, thereby diverging from the DP mechanisms applied in [7, 30, 57, 62], and advancing the field of privacy-preserving data publication.

While DP can offer worst-case privacy assurances [39], most real-world datasets do not contain the worst-case data record. Importantly, the prime concern for data owners and regulatory bodies is gauging the actual information leakage, as determined by real-world privacy attack simulations. Several privacy metrics [18, 35, 58] and attack strategies [8, 22, 24, 52] aimed at synthetic datasets currently exist. Nevertheless, these metrics don't align with DP principles, and many of the attack strategies make assumptions about reference or auxiliary data. In light of these issues, this paper introduces an innovative auditing scheme. The proposed scheme aligns with DP principles and draws inspiration from privacy games [60] and shadow models [46], allowing for a robust assessment of information leakage for any given data record. Moreover, we experimentally show that the proposed auditing scheme is more robust than current attacks [8, 22, 24, 52] that target synthetic datasets. Readers can refer to Appendix A.1 for a detailed literature review.

In summary, current literature offers several methodologies to address vertically partitioned data publication, but they come with notable shortcomings. In this paper, we propose VFLGAN as an effective solution to mitigate those shortcomings. Additionally, recognizing the privacy risks inherent in synthetic datasets, we incorporate a differentially private mechanism to ensure that VFLGAN satisfies a DP guarantee. While several mechanisms exist to provide centralized GANs with DP guarantees, as discussed in Appendix A.1.2, these mechanisms are unsuitable for VFLGAN due to the shared discriminator. To address this, we design a variant of the Gaussian mechanism tailored for DP-VFLGAN. Last, it is important to note that DP provides a privacy guarantee for the worst case, which rarely (if not never) exists in real-world datasets. Therefore, a practical auditing scheme is required to accurately estimate the privacy risk of training data. Existing privacy leakage measurements and attacks, which do not adhere to DP principles or rest on unrealistic assumptions, are inadequate for this purpose. In this paper, we design a novel and practical auditing scheme to effectively estimate the privacy risk of any given record.

1.1 Contributions

This paper outlines the following contributions to address the aforementioned research gaps.

- Through comprehensive investigation, we identified a critical limitation in a recently proposed GAN-based method for vertically partitioned data publication (presented at PETS 2023 [29]). Specifically, this model fails to effectively learn the correlation among attributes across different parties.
- We introduce the *first* Vertical Federated Learning (VFL)-based Generative Adversarial Network, named VFLGAN. This novel model is adept at learning correlations among attributes between different parties (as demonstrated in Fig. 1) and is equipped to handle both continuous and categorical attributes efficiently.
- A new Gaussian mechanism has been developed, equipping DP-VFLGAN with a (ϵ, δ)-Differential Privacy (DP) guarantee. This enhancement ensures heightened privacy protection in data publication.
- We propose a pragmatic auditing scheme - a privacy leakage measurement - that operates without reliance on unrealistic assumptions. This scheme quantitatively assesses the privacy risk of synthetic datasets.
- Extensive experiments were conducted to evaluate the quality of synthetic datasets generated by VFLGAN rigorously. We applied multiple metrics for a thorough assessment. Furthermore, the effectiveness of the proposed Gaussian mechanism was also evaluated using our innovative auditing method, demonstrating its efficacy in practical scenarios.
- The code[1] will be released at publication.

Readers can refer to **Appendix** A.8 for a summary of notation.

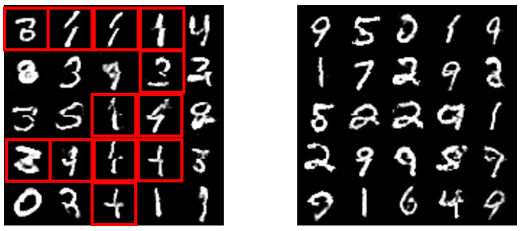


Figure 1: This figure shows synthetic samples generated by GANs trained on vertically partitioned MNIST data (the digits are split evenly into upper and lower halves). The left figure displays samples generated by VertiGAN [29], highlighting unrecognizable and discontinuous digits. The right figure shows samples generated by the proposed VFLGAN.

2 PRELIMINERIES

This section provides preliminaries of DP and GANs to facilitate a comprehensive understanding of the proposed VFLGAN. Besides, the training process of VertiGAN is introduced briefly.

2.1 Differential Privacy

Differential privacy [14] provides a rigorous privacy guarantee that can be quantitatively analyzed. The pure ϵ-DP is defined as follows.

Definition 1. *(ϵ-DP). A randomized mechanism $f : D \to R$ satisfies ϵ-differential privacy (ϵ-DP) if for any adjacent $D, D' \in \mathcal{D}$ and $S \subset R$*

$$Pr[f(D) \in S] \leq e^{\epsilon} Pr[f(D') \in S].$$

In the literature, the most commonly used DP is a relaxed version of the pure DP, which allows the mechanism to satisfy ϵ-DP most of the time but not satisfy ϵ-DP with a small probability, δ. The relaxed version, (ϵ, δ)-DP [15], is defined as follows.

[1] https://github.com/YuanXun2024/VFLGAN

Definition 2. *((ϵ, δ)-DP). A randomized mechanism $f : D \to R$ provides (ϵ, δ)-differential privacy ((ϵ, δ)-DP) if for any adjacent $D, D' \in \mathcal{D}$ and $S \subset R$*

$$Pr[f(D) \in S] \leq e^{\epsilon} Pr[f(D') \in S] + \delta.$$

In [40], the α-Rényi divergences between $f(D)$ and $f(D')$ are applied to define Rényi Differential Privacy (RDP) which is a generalization of differential privacy. $(\alpha, \epsilon(\alpha))$-RDP is defined as follows.

Definition 3. *($(\alpha, \epsilon(\alpha))$-RDP). A randomized mechanism $f : D \to R$ is said to have $\epsilon(\alpha)$-Rényi differential privacy of order α, or $(\alpha, \epsilon(\alpha))$-RDP for short if for any adjacent $D, D' \in \mathcal{D}$ it holds that*

$$D_{\alpha}\left(f(D) \| f\left(D'\right)\right) = \frac{1}{\alpha-1} \log \mathbb{E}_{x \sim f(D)} \left[\left(\frac{\Pr[f(D)=x]}{\Pr[f(D')=x]} \right)^{\alpha-1} \right] \leq \epsilon.$$

[40] proves that the Gaussian mechanism can guarantee an RDP.

Proposition 1. *(Gaussian Mechanism) Let $f : D \to R$ be an arbitrary function with sensitivity being*

$$\Delta_2 f = \max_{D,D'} \|f(D) - f\left(D'\right)\|_2$$

for any adjacent $D, D' \in \mathcal{D}$. The Gaussian Mechanism M_{σ},

$$\mathcal{M}_{\sigma}(\boldsymbol{x}) = f(\boldsymbol{x}) + \mathcal{N}\left(0, \sigma^2 I\right)$$

provides $\left(\alpha, \alpha \Delta_2 f^2 / 2\sigma^2\right)$-RDP.

The (R)DP budget should be accumulated if we apply multiple mechanisms to process the data sequentially as we train deep learning (DL) models for multiple iterations. We can calculate the accumulated RDP budget by the following proposition [40].

Proposition 2. *(Composition of RDP) Let $f : D \to R_1$ be (α, ϵ_1)-RDP and $g : R_1 \times D \to R_2$ be (α, ϵ_2)-RDP, then the mechanism defined as (X, Y), where $X \sim f(D)$ and $Y \sim g(X, D)$, satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

According to the following proposition, RDP can be converted to (ϵ, δ)-DP and the proof can be found in [40].

Proposition 3. *(From RDP to (ϵ, δ)-DP) If f is an $(\alpha, \epsilon(\alpha))$-RDP mechanism, it also satisfies $(\epsilon(\alpha) + \frac{log1/\delta}{\alpha-1}, \delta)$-DP for any $0 < \delta < 1$.*

According to Proposition 3, given a δ we can get a tight (ϵ', δ)-DP bound by

$$\epsilon' = \min_{\alpha}(\epsilon(\alpha) + \frac{log1/\delta}{\alpha - 1}). \tag{1}$$

[54] provides a tight upper bound on RDP by considering the combination of the subsampling procedure and random mechanism. This is important for differentially private DL since DL models are mostly updated according to a subsampled mini-batch of data. The enhanced RDP bound can be calculated according to the following proposition, and the proof can be found in [54].

Proposition 4. *(RDP for Subsampled Mechanisms). Given a dataset of n points drawn from a domain $\mathcal{X}$ and a (randomized) mechanism $\mathcal{M}$ that takes an input from $\mathcal{X}^m$ for $m \leq n$, let the randomized algorithm $\mathcal{M}\circ$ subsample be defined as (1) subsample: subsample without replacement m datapoints of the dataset (sampling rate $\gamma = m/n$), and (2) apply $\mathcal{M}$: a randomized algorithm taking the subsampled dataset as the input. For all integers $\alpha \geq 2$, if $\mathcal{M}$ obeys $(\alpha, \epsilon(\alpha))$-RDP, then this new randomized algorithm $\mathcal{M}\circ$ subsample obeys $(\alpha, \epsilon'(\alpha))$-RDP where,*

$$\begin{aligned}\epsilon'(\alpha) \leq & \frac{1}{\alpha-1} \log \left(1 + \gamma^2 \binom{\alpha}{2} \min \right. \\ & \left\{4\left(e^{\epsilon(2)} - 1\right), e^{\epsilon(2)} \min\left\{2, \left(e^{\epsilon(\infty)} - 1\right)^2\right\}\right\} \\ & \left. + \sum_{j=3}^{\alpha} \gamma^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min\left\{2, \left(e^{\epsilon(\infty)} - 1\right)^j\right\}\right)\end{aligned}$$

Last, the following post-processing theorem [16] is convenient, with which we can say a framework satisfies DP or RDP if any intermediate function of the framework satisfies DP or RDP.

Proposition 5. *(Post-processing). If $f(\cdot)$ satisfies (ϵ, δ)-DP, $g(f(\cdot))$ will satisfy (ϵ, δ)-DP for any function $g(\cdot)$. Similarly, if $f(\cdot)$ satisfies (α, ϵ)-RDP, $g(f(\cdot))$ will satisfy (α, ϵ)-RDP for any function $g(\cdot)$.*

2.2 Generative Adversarial Models

Given a dataset X where the data record $x \in X$ follows the distribution P, the generator of GAN, G, aims to generate synthetic data $\tilde{x}$, $\tilde{x} = G(z)$, that follows the distribution $P_{G(z)}$ similar to P. The input z is sampled from a simple distribution, such as the uniform distribution or a Gaussian distribution. The above object can be achieved with the help of a discriminator, D. The generator and discriminator are trained through a competing game, where the discriminator is trained to distinguish x and $\tilde{x}$, and the generator is trained to generate high-quality $\tilde{x}$ to fool the discriminator. The game between the generator and the discriminator can be formally expressed as the following min-max objective [19],

$$\min_{G} \max_{D} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim P_{G(z)}}[\log(1 - D(\tilde{\boldsymbol{x}}))]. \tag{2}$$

As proved in [19], this objective leads to minimizing the Jensen-Shannon divergence between P and $P_{G(z)}$. However, the training process of optimizing the objective (2) is unstable due to discriminator saturating, which results in vanishing gradients. [2] pointed out that the Jensen–Shannon divergence is not continuous and does not provide usable gradients and proposed a new objective, i.e., minimizing the Wasserstein-1 distance between P and $P_{G(z)}$, which is continuous everywhere and differentiable almost everywhere under mild assumptions. Based on the new objective, Wasserstein GAN (WGAN) is proposed in [2]. Following this idea, subsequent works [21, 55, 56] propose variants of WGAN to improve the quality of generated data. Same as the previous works [7, 29], we adopt the optimization objectives of WGAN_GP [21] for the proposed VFLGAN as,

$$\begin{aligned}&\min_{D} -\mathbb{E}[D(\boldsymbol{x})] + \mathbb{E}[D(\tilde{\boldsymbol{x}})] + \lambda \mathbb{E}\left[\left(\|\nabla D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2\right], \\ &\max_{G} \mathbb{E}[D(\tilde{\boldsymbol{x}})],\end{aligned} \tag{3}$$

where $\hat{\boldsymbol{x}} = \beta \boldsymbol{x} + (1 - \beta)\tilde{\boldsymbol{x}}$ and $\beta \sim \mathbb{U}(0, 1)$.

2.3 VertiGAN

Figure 2 shows the training framework of VertiGAN for the two-party scenario. Each party shares the same generator backbone (G_b). Party i maintains a private generator head (G_{hi}) and a private discriminator (D_i). The local update in each party is the same process as WGAN_GP [21], except for the update of the generator

backbone (G_b). Thus, same as [21], the optimization objective of party $i \in \{1, 2\}$ can be expressed as,

$$\begin{aligned} &\min_{D_i} -\mathbb{E}[D_i(\boldsymbol{x_i})] + \mathbb{E}[D_i(\tilde{\boldsymbol{x}}_i)] + \lambda \mathbb{E}\left[(\|\nabla D_i(\hat{\boldsymbol{x}}_i)\|_2 - 1)^2\right], \\ &\max_{G_i} \mathbb{E}[D_i(\tilde{\boldsymbol{x}}_i)], \end{aligned} \tag{4}$$

where $\hat{\boldsymbol{x}}_i = \beta \boldsymbol{x}_i + (1-\beta)\tilde{\boldsymbol{x}}_i$ and $\beta \sim \mathbb{U}(0, 1)$. For the update of the generator backbone (G_b), each party sends the local gradients of G_b, i.e., $\mathcal{G}^i_{G_b}$, to the server. The server summarizes the local gradients as,

$$\mathcal{G}_{G_b} = \mathcal{G}^1_{G_b} + \mathcal{G}^2_{G_b}, \tag{5}$$

and sends $\mathcal{G}_{G_b}$ to the local parties. Then, the local party updates the local G_b with $\mathcal{G}_{G_b}$. Equation (5) is the main idea of horizontal federated learning (HFL).

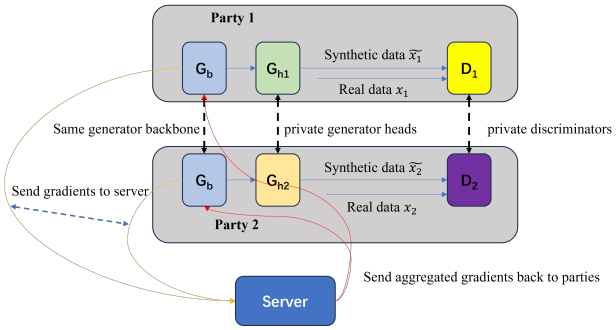


Figure 2: This figure shows the framework of VertiGAN.

3 PROPOSED VERTICAL FEDERATED LEARNING-BASED GAN

This section first introduces our system model. According to the system model, we formulate the vertically partitioned data publication problem as a min-max optimization problem to train the proposed VFLGAN. Subsequently, we specify the architecture of the proposed VFLGAN for the two-party case. Following the architectural overview, the training process of VFLGAN is described in detail. Then, we introduce the differentially private version of VFLGAN, i.e., DP-VFLGAN. The section concludes by delineating the differences between VertiGAN and VFLGAN.

3.1 System Model and Problem Formulation

Our system model considers a similar scenario as discussed in [29], where there are M non-colluding parties. Each party P_i maintains a private dataset $X_i \in \mathbb{R}^{N \times |A_i|}$ with N records where A_i denotes the attribute set of X_i. Now, we consider the following assumptions for our proposed system model.

Assumption 1 Records in different datasets X_i with the same index belong to the same object, which can be achieved by applying private set intersection protocols [9, 26] in practice. To facilitate the alignment of training inputs across parties without direct data sharing, a pseudorandom number generator can be employed during the training process.

Assumption 2 There is no common attribute among the parties.

Assumption 3 The local parties and the central server are honest but curious, i.e., correctly follow the protocols but try to infer sensitive information from other parties.

With the above assumptions, M private datasets can be combined to construct a new dataset X, i.e., $X = [X_1, X_2, \cdots, X_M] \in \mathbb{R}^{N \times \sum_{i=1}^{M} |A_i|}$, where $[\cdots]$ denotes a concatenation function. The parties aim to generate a synthetic dataset $\tilde{X}$ in which each record $\tilde{\boldsymbol{x}} \in \tilde{X}$ follows a similar distribution to that of $\boldsymbol{x} \in X$,

$$P_{\tilde{\boldsymbol{x}}} \approx P_{\boldsymbol{x}}, \tag{6}$$

while keeping the local data secret from other parties.

Here, we use GAN-based generators to generate synthetic records that satisfy (6). Figure 3 illustrates our system model in detail. There is a private dataset X_i, a local generator G_i, and a local discriminator D_i in each party P_i. The server maintains a shared discriminator D_s and is responsible for combining the synthetic data from all the parties. Solid lines between the server and parties represent the communication during the generalisation of the synthetic dataset (inference period), and the dashed lines represent the communication during the training process. In our system model, the private dataset X_i can only be accessed by the corresponding discriminator D_i. The shared discriminator D_s aims to guide the local generators to learn the correlation among attributes of different parties. During the training process, each D_i sends its intermediate feature to D_s and D_s return gradients to update D_i and G_i. During the inference process, the local generator G_i generates partial synthetic records, $\tilde{\boldsymbol{x}}_i = G_i(\boldsymbol{z})$, where $\boldsymbol{z}$ is the same for all parties achieved by a pseudorandom number generator at each local party. Then, the partial synthetic records are concatenated in the server to get a complete synthetic record, $\tilde{x} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \cdots, \tilde{\boldsymbol{x}}_M]$. According to our system model, the objective (6) can be expressed as the following,

$$P_{\tilde{\boldsymbol{x}}} \approx P_{\boldsymbol{x}}, \quad \tilde{\boldsymbol{x}} = [G_1(\boldsymbol{z}), G_2(\boldsymbol{z}), \cdots, G_M(\boldsymbol{z})]. \tag{7}$$

Thus, the problem targeted by this paper is to train M generators that satisfy (7). As mentioned in Section 2.2, this problem can be transformed into a min-max optimization problem to obtain such generators. According to the optimization objectives of WGAN_GP (4), the vertically partitioned data publication problem of our system model can be formulated as,

$$\min_{D_1, \cdots, D_M, D_s} \sum_{i=1}^{M} \mathcal{L}(D_i, \tilde{\boldsymbol{x}}_i, \boldsymbol{x}_i) + \lambda_1 \mathcal{L}(D_s, \tilde{\boldsymbol{f}}, \boldsymbol{f}), \tag{8}$$

$$\max_{G_1, \cdots, G_M} \sum_{i=1}^{M} \mathbb{E}[D_i(\tilde{\boldsymbol{x}}_i)] + \lambda_2 \mathbb{E}[D_s(\tilde{\boldsymbol{f}})], \tag{9}$$

where $\mathcal{L}(D, \tilde{\boldsymbol{x}}, \boldsymbol{x}) \triangleq -\mathbb{E}[D(\boldsymbol{x})] + \mathbb{E}[D(\tilde{\boldsymbol{x}})] + \lambda \mathbb{E}\left[(\|\nabla D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]$, $\tilde{\boldsymbol{f}}$ and $\boldsymbol{f}$ are the concatenations of the intermediate features of D_1 to D_M when the inputs are synthetic data and real data respectively, and λ_1 and λ_2 are balancing coefficients.

3.2 Overview of VFLGAN for Two-party Case

Figure 4 illustrates the proposed VFLGAN for the two-party scenario. There are two private generators, G_1 and G_2, two private discriminators, D_1 and D_2, and one shared discriminator, D_s, in the framework. The generators produce synthetic data with a vector of Gaussian noise $\boldsymbol{z}$, i.e., $\tilde{\boldsymbol{x}}_i = G_i(\boldsymbol{z})$. The discriminators, D_i where

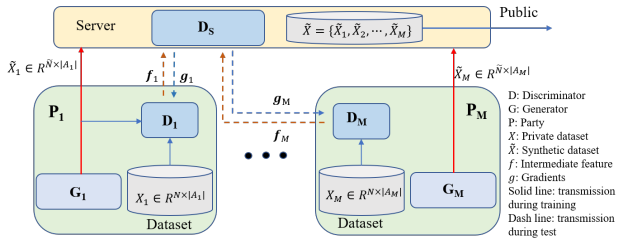


Figure 3: System Model.

$i \in \{1, 2\}$, are trained to distinguish synthetic data $\tilde{\boldsymbol{x}}_i$ and real data $\boldsymbol{x}_i$. Thus, the gradients of D_i on $\tilde{\boldsymbol{x}}_i$ can guide G_i to generate synthetic data with a similar distribution to the real data $\boldsymbol{x}_i$, i.e.,

$$P_{G_i(\boldsymbol{z})} \approx P_{\boldsymbol{x}_i}. \tag{10}$$

Figure 11 in the Appendix shows the detailed structure of the VFLGAN discriminators. In D_i $i \in \{1, 2\}$, the layers before the intermediate feature construct the first part of D_i in Fig. 4, and the layers after the intermediate feature construct the second part of D_i. The intermediate feature ($\boldsymbol{f}_i$) is the output of the first part of D_i, i.e.,

$$\boldsymbol{f}_i = D_i^1(\boldsymbol{x}_i) \quad i \in \{1, 2\}, \tag{11}$$

where D_i^1 denotes the first part of D_i and $\boldsymbol{x}_i$ denotes the input of D_i. The intermediate features are transmitted to the server where the concatenation of the intermediate features is input to D_s. D_s is trained to distinguish $[\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2]$ and $[\boldsymbol{x}_1, \boldsymbol{x}_2]$ by optimizing $\mathcal{L}(D_s, \tilde{\boldsymbol{f}}, \boldsymbol{f})$ in (8). As a result, by optimizing (9), D_s can guide G_1 and G_2 to learn the correlation between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and generate better synthetic data, i.e.,

$$P_{[G_1(\boldsymbol{z}), G_2(\boldsymbol{z})]} \approx P_{[\boldsymbol{x}_1, \boldsymbol{x}_2]}. \tag{12}$$

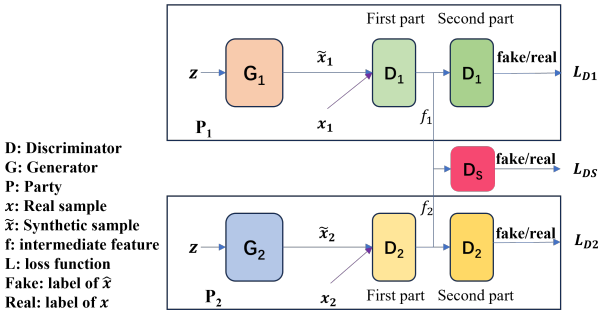


Figure 4: Framework of the proposed VFLGAN.

Concerns may arise regarding the potential for privacy leakage through intermediate features within a neural network. However, reconstructing the input from the neural network's output, without access to the model's parameters, presents a difficult ill-posed problem [13]. For instance, the authors in [59] attempt to invert neural networks used for classification, where the adversary is assumed to possess an auxiliary dataset and the ability to interact with the model by submitting requests and receiving responses. Nevertheless, in our scenario, such access to the model or the ability to submit requests is precluded. Another study [3] endeavours to reconstruct training data from the outputs of a trained neural network, with the adversary having access to all but one record in the training dataset and white-box access to the model. Despite the strong assumptions made in [3, 59], the reconstructed images are notably blurred, rendering them less precise than the original images. While a blurred image might still reveal some private information, a blurred tabular record is likely to disclose minimal privacy details. Furthermore, to address these concerns, we have developed an auditing scheme, detailed in Section 4.2, designed to assess the extent of privacy leakage through intermediate features.

3.3 Training Process of VFLGAN

Following the training procedure of previous works [19, 21], we optimize the discriminators and generators in sequence, i.e., we first optimize the discriminators for T_d iterations and then optimize the generators for one iteration. According to our optimization objective (8), the loss function of D_s is $\mathcal{L}(D_s, \tilde{\boldsymbol{f}}, \boldsymbol{f})$ and the loss function for D_1 and D_2 is $\mathcal{L}(D_i, \tilde{\boldsymbol{x}}_i, \boldsymbol{x}_i)$. In the remaining paper, we use $\mathcal{L}_{D_s}$ and $\mathcal{L}_{D_i}$ to denote the above loss functions for abbreviation. The gradients of D_s parameters can be calculated according to its loss function by,

$$\mathcal{G}_{D_s} = \nabla_{\boldsymbol{\theta}_{D_s}} \mathcal{L}_{D_s}. \tag{13}$$

Note that $\mathcal{L}_{D_s}$ also contributes to the gradients of the first part of D_1 and D_2 parameters. Thus, the gradients of D_1 and D_2 parameter can be calculated by,

$$\mathcal{G}_{D_1} = \nabla_{\boldsymbol{\theta}_{D_1}} \mathcal{L}_{D_1} + \nabla_{\boldsymbol{\theta}_{D_1}} \mathcal{L}_{D_s}, \tag{14}$$

$$\mathcal{G}_{D_2} = \nabla_{\boldsymbol{\theta}_{D_2}} \mathcal{L}_{D_2} + \nabla_{\boldsymbol{\theta}_{D_2}} \mathcal{L}_{D_s}. \tag{15}$$

Finally, the parameters of the discriminators can be updated by,

$$\boldsymbol{\theta}_{D_i} = \boldsymbol{\theta}_{D_i} - \eta_{D_i} \mathcal{G}_{D_i} \quad i \in \{1, 2, S\}, \tag{16}$$

where η_{D_i} denotes the learning rate of D_i.

Notably, when we optimize the generators according to the objective function (9), the output of the discriminators is maximised. However, the optimizers in the DL field, like Adam [31], usually minimize the loss function. Thus, the loss function for the generators is derived as,

$$\mathcal{L}_G = -\sum_{i=1}^{2} \mathbb{E}[D_i(G_i(\boldsymbol{z})] - \mathbb{E}[D_s([\boldsymbol{f}_1, \boldsymbol{f}_2])], \tag{17}$$

where $\boldsymbol{f}_i$ denotes the intermediate feature of D_i as shown in Fig. 4. Then, the gradients and parameter update of G_i can be calculated as follows,

$$\mathcal{G}_{G_i} = \nabla_{\boldsymbol{\theta}_{G_i}} \mathcal{L}_G, \tag{18}$$

$$\boldsymbol{\theta}_{G_i} = \boldsymbol{\theta}_{G_i} - \eta_{G_i} \mathcal{G}_{G_i} \quad i \in \{1, 2\}. \tag{19}$$

Note D_s also contributes to the loss function of G_1 and G_2, through which G_1 and G_2 learn to generate synthetic data satisfying (12).

The training process of the VFLGAN is summarized in Algorithm 1. First, we train the discriminators for T_d iterations. During each iteration, we subsample a mini-batch of real data, $[\boldsymbol{x}_1^B, \boldsymbol{x}_2^B]$, where B denotes the batch size, from the training dataset and generate a mini-batch of synthetic data $[\tilde{\boldsymbol{x}}_1^B, \tilde{\boldsymbol{x}}_2^B]$ with G_1 and G_2. And the discriminators are trained to distinguish real data and synthetic

data. Second, we train the generators to generate more realistic synthetic data for one iteration. The first and second steps repeat for T_{max} epochs, and the algorithm outputs trained G_1 and G_2.

Algorithm 1: Training Process of (DP-)VFLGAN

Input: G_1 and G_2: generators; D_1 and D_2: discriminators; $\boldsymbol{f}_i$: the intermediate feature of D_i; n_{D_1} and n_{D_2}: number of layers of D_1 and D_2; D_s: shared discriminator; X: dataset B: batch size; T_{max}: maximum training epochs; T_d: discriminators' update steps; η: learning rate; l: latent dimension; θ: parameters of VFLGAN; $\mathcal{G}^1_{D_i}$: gradients of parameters of the first layer of D_i.

Output: Trained G_1 and G_2.

Initialize the generators and discriminators;
for *epoch in* $\{1, 2, \cdots, T_{max}\}$ **do**
  // update discriminators (line 1 to 19);
  for $iter$ *in* $\{1, 2, \cdots, T_d\}$ **do**
    $\boldsymbol{x}^B = [\boldsymbol{x}^B_1, \boldsymbol{x}^B_2] \subset X$ *// Subsample a mini-batch of data;*
    Generate $\boldsymbol{z}^B$ with $z \sim \mathcal{N}(0,1)^l$
    $\tilde{\boldsymbol{x}}^B = [\tilde{\boldsymbol{x}}^B_1, \tilde{\boldsymbol{x}}^B_2] = [G_1(\boldsymbol{z}^B), G_2(\boldsymbol{z}^B)]$ *// Generate synthetic data;*
    // Compute losses of D_1, D_2, and D_s (line 5 to 7);
    $\mathcal{L}_{D_1} = -\mathbb{E}[D_1(\boldsymbol{x}^B_1)] + \mathbb{E}[D_1(\tilde{\boldsymbol{x}}^B_1)]$
    $\quad + \lambda \mathbb{E}\left[\left(\|\nabla D_1(\alpha^B \boldsymbol{x}^B_1 + (1-\alpha^B)\tilde{\boldsymbol{x}}^B_1)\|_2 - 1\right)^2\right]$
    $\mathcal{L}_{D_2} = -\mathbb{E}[D_2(\boldsymbol{x}^B_2)] + \mathbb{E}[D_2(\tilde{\boldsymbol{x}}^B_2)]$
    $\quad + \lambda \mathbb{E}\left[\left(\|\nabla D_2(\alpha^B \boldsymbol{x}^B_2 + (1-\alpha^B)\tilde{\boldsymbol{x}}^B_2)\|_2 - 1\right)^2\right]$
    $\mathcal{L}_{D_s} = -\mathbb{E}[D_s([\boldsymbol{f}_1, \boldsymbol{f}_2])] + \mathbb{E}[D_s([\tilde{\boldsymbol{f}}_1, \tilde{\boldsymbol{f}}_2])]$
    $\quad + \lambda \mathbb{E}\left[\left(\|\nabla D_s(\alpha^B [\boldsymbol{f}_1, \boldsymbol{f}_2] + (1-\alpha^B)[\tilde{\boldsymbol{f}}_1, \tilde{\boldsymbol{f}}_2])\|_2 - 1\right)^2\right]$
    $\mathcal{G}_{D_1} = \nabla_{\boldsymbol{\theta}_{D_1}} \mathcal{L}_{D_1} + \nabla_{\boldsymbol{\theta}_{D_1}} \mathcal{L}_{D_s}$ *// Compute gradients of D_1;*
    $\mathcal{G}_{D_2} = \nabla_{\boldsymbol{\theta}_{D_2}} \mathcal{L}_{D_2} + \nabla_{\boldsymbol{\theta}_{D_2}} \mathcal{L}_{D_s}$ *// Compute gradients of D_2;*
    $\mathcal{G}_{D_s} = \nabla_{\boldsymbol{\theta}_{D_s}} \mathcal{L}_{D_s}$ *// Compute gradients of D_s;*
    $\boldsymbol{\theta}_{D_s} = \boldsymbol{\theta}_{D_s} - \eta_{D_s} \mathcal{G}_{D_s}$ *// Update parameters of D_s;*
    // Update parameters of the second to the last layers of D_1;
    $\boldsymbol{\theta}^{2:n_{D_1}}_{D_1} = \boldsymbol{\theta}^{2:n_{D_1}}_{D_1} - \eta_{D_1} \mathcal{G}^{2:n_{D_1}}_{D_1}$
    // Update parameters of the second to the last layers of D_2;
    $\boldsymbol{\theta}^{2:n_{D_2}}_{D_2} = \boldsymbol{\theta}^{2:n_{D_2}}_{D_2} - \eta_{D_2} \mathcal{G}^{2:n_{D_2}}_{D_2}$
    // Update parameters of the first layer of D_1 and D_2 (line 14 to 19);
    if *Training a differentially private version* **then**
      $\boldsymbol{\theta}^1_{D_1} = \boldsymbol{\theta}^1_{D_1} - \eta_{D_1}(clip(\mathcal{G}^1_{D_1}, C) + \mathcal{N}(0, \sigma^2(2C)^2 I))$
      $\boldsymbol{\theta}^1_{D_2} = \boldsymbol{\theta}^1_{D_2} - \eta_{D_2}(clip(\mathcal{G}^1_{D_2}, C) + \mathcal{N}(0, \sigma^2(2C)^2 I))$
    else
      $\boldsymbol{\theta}^1_{D_1} = \boldsymbol{\theta}^1_{D_1} - \eta_{D_1} \mathcal{G}^1_{D_1}$
      $\boldsymbol{\theta}^1_{D_2} = \boldsymbol{\theta}^1_{D_2} - \eta_{D_2} \mathcal{G}^1_{D_2}$
  // update generators (line 20 to 25);
  $\tilde{\boldsymbol{x}}^B = [G_1(\boldsymbol{z}^B), G_2(\boldsymbol{z}^B)]$ *// Generate a mini-batch of fake data $\tilde{\boldsymbol{x}}^B$;*
  $\mathcal{L}_{G_1} = -\mathbb{E}[D_1(\tilde{\boldsymbol{x}}_1)] - \mathbb{E}[D_s([\tilde{\boldsymbol{f}}_1, \tilde{\boldsymbol{f}}_2])]$ *// Compute losses of G_1;*
  $\mathcal{L}_{G_2} = -\mathbb{E}[D_2(\tilde{\boldsymbol{x}}_2)] - \mathbb{E}[D_s([\tilde{\boldsymbol{f}}_1, \tilde{\boldsymbol{f}}_2])]$ *// Compute losses of G_2;*
  where $\tilde{\boldsymbol{f}}_i$ is the intermediate feature when the input of D_i is $\tilde{\boldsymbol{x}}_i$
  // Compute gradients of G_1 and G_2;
  $\mathcal{G}_{G_1} = \nabla_{\boldsymbol{\theta}_{G_1}} \mathcal{L}_{G_1}, \quad \mathcal{G}_{G_2} = \nabla_{\boldsymbol{\theta}_{G_2}} \mathcal{L}_{G_2}$
  // Update parameters of G_1 and G_2;
  $\boldsymbol{\theta}_{G_1} = \boldsymbol{\theta}_{G_1} - \eta_{G_1} \mathcal{G}_{G_1}, \quad \boldsymbol{\theta}_{G_2} = \boldsymbol{\theta}_{G_2} - \eta_{G_2} \mathcal{G}_{G_2}$
Return G_1 and G_2.

3.4 Differentially Private VFLGAN

The training process of DP-VFLGAN is also summarized in Algorithm 1 since the training processes of VFLGAN and DP-VFLGAN are the same for most steps. There are mainly two differences between the two training processes. (i) We apply the proposed Gaussian mechanism (21) in DP-VFLGAN. When updating the parameters of the first linear layers of D_1 and D_2, we clip and add Gaussian noise to the gradients as follows,

$$clip(\mathcal{G}^1_{D_i}, C) = \mathcal{G}^1_{D_i} / \max\left(1, \left\|\mathcal{G}^1_{D_i}\right\|_2 / C\right) \quad i \in \{1, 2\}, \tag{20}$$

$$\mathcal{G}^1_{D_i} = clip(\mathcal{G}^1_{D_i}, C) + \mathcal{N}\left(0, \sigma^2(2C)^2 I\right), \tag{21}$$

where $\mathcal{G}^1_{D_i}$ denotes the gradients of the first layer parameters of D_i and C denotes the clipping bound. Previous Gaussian mechanisms for GANs [29, 57, 62] clip and add Gaussian noise to all discriminator parameters. However, we proved in Theorem 1 that clipping and adding Gaussian noise to the gradients of the first-layer parameters of D_1 and D_2 can provide a DP guarantee for G_1 and G_2. (ii) We need to set a privacy budget before training DP-VFLGAN to ensure G_1 and G_2 satisfy (ϵ, δ)-DP. Here, we apply the official implementation of [54] to select the proper T_{max} and σ to achieve the privacy budget, i.e., (ϵ, δ)-DP, and the details are described in Section 3.5.

3.5 RDP Guarantee and Proof

Theorem 1. *(RDP Guarantee) All the local discriminators and generators satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP in one training iteration of DP-VFLGAN.*

Proof. The proof of Theorem 1 is presented in **Appendix** A.2. □

Now, we introduce how to select proper σ and T_{max} to meet our DP budget using Theorem 1. First, the RDP guarantee in Theorem 1 can be enhanced by Proposition 4 for external attackers and the server since we subsample mini-batch records from the whole training dataset. Then, RDP budget is accumulated by T_{max} iterations, which can be calculated according to Proposition 2. Last, the RDP guarantee is converted to DP guarantee using (1). We can adjust the σ and T_{max} to make the calculated DP guarantee meet our DP budget. Notably, similar to the DP guarantee in [29], the above DP guarantee specifically addresses external threats. This is due to the deterministic nature of mini-batch selection in each training iteration for internal parties (excluding the server), as opposed to a subsampling process. For internal adversaries, privacy can be enhanced by sacrificing efficiency. For example, the mini-batch size can be changed to $\hat{B} > B$. When updating the parameters, each party can randomly select the gradients of B samples and mask the gradients of other $(\hat{B} - B)$ samples. In this way, all parties do not know precisely which samples are used by others to update the parameters.

3.6 Differences between VFLGAN and VertiGAN

As discussed in Section 2.3, in VertiGAN, the objective functions (4) of the local discriminator and generator in each party are the same as WGAN_GP's objective functions (2). According to [21], this can ensure that the synthetic data $(\tilde{\boldsymbol{x}}_i)$ generated by part i follows a similar distribution of the real data $(\boldsymbol{x}_i)$ of party i, i.e., $P_{\tilde{\boldsymbol{x}}_i} \simeq P_{\boldsymbol{x}_i}$. After calculating the gradients of each local generator backbone, VertiGAN applies HFL (5) to maintain the same generator backbone across all parties and assumes that the same generator backbones can learn the correlations among attributes across those

parties, i.e., $P_{[\tilde{\boldsymbol{x}}_1,\cdots,\tilde{\boldsymbol{x}}_N]} \simeq P_{[\tilde{\boldsymbol{x}}_1,\cdots,\tilde{\boldsymbol{x}}_N]}$, which is experimentally proved less effective in this paper. On the other hand, VFLGAN incorporates the objective functions of WGAN_GP into the VFL framework with a shared discriminator. Accordingly, we propose novel objective functions, (8) and (9), for VFLGAN. From (8) and (9), we can see that the objective functions of VFLGAN involve all parties through the shared discriminator while the objective functions of each party are independent in VertiGAN. In (8) and (9), according to [21], the fractions involving local discriminatory D_i can ensure that $P_{\tilde{\boldsymbol{x}}_i} \simeq P_{\boldsymbol{x}_i}$ and the fractions involving the shared discriminator D_s can ensure that $P_{[\tilde{\boldsymbol{x}}_1,\cdots,\tilde{\boldsymbol{x}}_N]} \simeq P_{[\tilde{\boldsymbol{x}}_1,\cdots,\tilde{\boldsymbol{x}}_N]}$.

Another difference is that the differentially private VertiGAN applies the same Gaussian mechanism as [57, 62], which clips and adds noise to all gradients. However, as proved in Thereom 1, clipping and adding noise to the gradients of the first linear layer of the local discriminator is enough to provide a DP guarantee to all local discriminators and local generators in DP-VFLGAN. Based on this discovery, DP-VFLGAN applies a new variant of the Gaussian mechanism as discussed in Section 3.4.

4 PRIVACY LEAKAGE MEASUREMENT

In this section, we introduce two distinct auditing schemes, i.e., the Auditing Scheme for Synthetic Datasets (ASSD) and the Auditing Scheme for Intermediate Features (ASIF). These schemes are designed to conduct MI attacks within a leave-one-out setting [60], a scenario in which the adversary is presumed to be aware of the entire dataset except for the target record (x_t). While stringent for an external adversary, this assumption provides an empirical upper limit on the success rates of MI attacks. Additionally, this setting is deemed realistic for data publishers who can readily manipulate the inclusion of the target record in the training process. The differentiation between our MI attack strategy and that described in [60] hinges on two primary aspects: (i) The focus of our MI attack is on synthetic datasets, as opposed to the machine learning models targeted in [60]; (ii) Our auditing schemes adopt the shadow model attack method [25, 46, 47] as the adversary model, contrasting with the binary hypothesis test approach utilized in [60], which is specifically crafted for classification models.

4.1 Auditing Scheme for Synthetic Datasets

The following privacy game can define the MI attack applied in ASSD. **(i)** First, the challenger selects a fixed target record $\boldsymbol{x}_t$ from a given training dataset X. **(ii)** Then, the challenger trains a generator $G_0 \xleftarrow{s_0} \mathcal{T}(X \setminus \boldsymbol{x}_t)$ on $X \setminus \boldsymbol{x}_t$ (dataset X excluding $\boldsymbol{x}_t$) by using a fresh random seed s_0 in the training algorithm $\mathcal{T}$. In this paper, $\mathcal{T}$ refers to Algorithm 1. **(iii)** The challenger trains another generator $G_1 \xleftarrow{s_1} \mathcal{T}(X)$ on X by using a fresh random seed s_1. **(iv)** The challenger applies the two generators, G_0 and G_1, to produce two synthetic datasets, $\tilde{X}_1$ and $\tilde{X}_2$. **(v)** The challenger flips a random unbiased coin $b \in \{0, 1\}$ and sends the synthetic dataset and target record $\{\tilde{X}_b, \boldsymbol{x}_t\}$ to the adversary. **(vi)** The adversary tries to figure out the true b based on the observation of $\{\tilde{X}_b, \boldsymbol{x}_t\}$, i.e., $\hat{b} \leftarrow \mathcal{A}(\tilde{X}_b, \boldsymbol{x}_t)$. **(vii)** If $\hat{b} = b$, the adversary wins. Otherwise, the challenger wins. This MI attack is summarized in Algorithm 2.

We apply the shadow modelling approach [46] to train the adversary model $\mathcal{A}$ in Algorithm 2 through the following process. **(i)** Train M generators, $G_{0_{1:M}}$, on dataset $X \setminus \boldsymbol{x}_t$ by using different random seeds $s_{0_{1:M}}$ in the training algorithm $\mathcal{T}$. **(ii)** Apply $G_{0_{1:M}}$ to generate M synthetic datasets $\tilde{X}_{0_{1:M}}$. **(iii)** Train M generators, $G_{1_{1:M}}$, on dataset X by using different random seeds $s_{1_{1:M}}$. **(vi)** Apply $G_{1_{1:M}}$ to generate M synthetic datasets $\tilde{X}_{1_{1:M}}$. **(v)** Extract features of the synthetic datasets $\tilde{X}_{0_{1:M}}$ and $\tilde{X}_{1_{1:M}}$. **(vi)** Train the adversary model $\mathcal{A}$ to distinguish whether the features are from $\tilde{X}_{0_{1:M}}$ or $\tilde{X}_{1_{1:M}}$. The training process is summarized in Algorithm 3.

In summary, the adversary is trained to detect whether the target exists in the training data through the features of the synthetic dataset. Then, we evaluate the capability of the adversary through the MI attack. The success rate of the MI attack can reflect the potential privacy leakage that external attackers can take advantage of through the synthetic dataset and MI attack.

Algorithm 2: Membership Inference Attack

Input: Training algorithm $\mathcal{T}$; dataset X; target record $\boldsymbol{x}_t$; unbiased coin b; fresh random seeds s_0 and s_1; generators of VFLGAN G_i; synthetic dataset $\tilde{X}$; Gaussian noise $\boldsymbol{z}$.
Output: Success or failure.

$G_0 \xleftarrow{s_0} \mathcal{T}(X \setminus \boldsymbol{x}_t)$
$G_1 \xleftarrow{s_1} \mathcal{T}(X)$
$\tilde{X}_0 \longleftarrow G_0(\boldsymbol{z})$ & $\tilde{X}_1 \longleftarrow G_1(\boldsymbol{z})$
$b \sim \{0, 1\}$
$\hat{b} \longleftarrow \mathcal{A}(\tilde{X}_b, \boldsymbol{x}_t)$
if $\hat{b} == b$ **then**
  Output success
else
  Output failure
end if

Algorithm 3: Training the Adversary of MI Attack

Input: Training algorithms $\mathcal{T}$ and $\mathcal{T}_{\mathcal{A}}$; dataset X; target record $\boldsymbol{x}_t$; random seeds $s_{0_{1:M}}$ and $s_{1_{1:M}}$; synthetic dataset $\tilde{X}$; feature extraction function $Extr(\cdot)$.
Output: Trained $\mathcal{A}$.

$G_{0_{1:M}} \xleftarrow{s_{0_{1:M}}} \mathcal{T}(X \setminus \boldsymbol{x}_t)$
$\tilde{X}_{0_{1:M}} \longleftarrow G_{0_{1:M}}$
$G_{1_{1:M}} \xleftarrow{s_{1_{1:M}}} \mathcal{T}(X)$
$\tilde{X}_{1_{1:M}} \longleftarrow G_{1_{1:M}}$
$Feat_{0_{1:M}} \longleftarrow Extr(\tilde{X}_{0_{1:M}})$ & $Feat_{1_{1:M}} \longleftarrow Extr(\tilde{X}_{1_{1:M}})$
$\mathcal{A} \longleftarrow \mathcal{T}_{\mathcal{A}}(Feat_{0_{1:M}}, Feat_{1_{1:M}})$

4.2 Auditing Scheme for Intermediate Features

The MI attack and training process of the adversary applied in ASIF are very similar to those (Algorithm 2 and Algorithm 3) of ASSD. So we briefly introduce ASIF here and details can be found in Appendix A.3. In ASIF, the challenger also trains two VFLGANs (θ_0 and θ_1) on datasets X and $X \setminus \boldsymbol{x}_t$, respectively, and uses their first part of local discriminators to generate intermediate features. The adversary tries to figure out whether the given intermediate feature is from θ_0 or θ_1. Similar to ASSD, ASIF applies shadow model attack as the adversary model. The difference is that the adversary model of ASSD is to figure out whether the target record $\boldsymbol{x}_t$ appears in the training data through a given synthetic dataset, while the adversary

model of ASIF is to do the same job through given intermediate features.

5 EXPERIMENTAL DETAILS AND RESULTS

In this section, we commence by detailing our experimental setup, including the datasets, baseline methods, and evaluation metrics. Following this, we rigorously assess the effectiveness of VFLGAN and its privacy-enhanced counterpart, DP-VFLGAN. The section concludes with an in-depth analysis of the privacy leakage associated with VFLGAN and DP-VFLGAN, employing our newly developed privacy auditing schemes, ASSD and ASIF.

5.1 Experiment Setup

5.1.1 Datasets. We employ three datasets for evaluation: the MNIST dataset, Adult dataset [5], Wine Quality datasets [11], Credit dataset [1], and HCV dataset [42]. All datasets utilized in this study are derived from real-world sources. To simulate the scenario where two distinct parties possess different attributes of the same group of data samples, we divided each dataset into two sub-datasets through a vertical split.

MNIST is a widely recognized dataset comprised of handwriting digit images. In our experiment, each image is transformed into a one-dimensional vector with 784 attributes. These attributes are split between two clients in a vertically partitioned manner: the first 392 attributes to Client 1, and the remaining 392 attributes to Client 2. Each attribute is an integer ranging from 0 to 255. Although image data is not typically partitioned vertically, we utilize this to demonstrate the shortcomings of VertiGAN [29], which our proposed VFLGAN addresses effectively.

Adult [5] is a prominent machine-learning dataset used for classification tasks. It consists of fifteen attributes: fourteen features describing various aspects of an individual, and a label indicating whether the person's income exceeds 50,000 dollars. The dataset comprises six integer attributes and nine categorical attributes.

Wine quality [11] is used for classification tasks and includes two subsets: Red-Wine-Quality and White-Wine-Quality. Each subset contains twelve attributes, with eleven continuous attributes serving as features of a wine sample and one categorical attribute denoting the wine's quality.

Credit [1] is a financial dataset used for classification tasks. It consists of 21 attributes: 20 features describing various aspects of an individual and a label indicating the person's credit. The dataset comprises 3 integer attributes and 18 categorical attributes.

HCV [42] is a medical dataset used for classification tasks. It consists of 29 attributes: 28 features describing personal information and medicine check results and a label indicating the HCV staging. The dataset comprises 13 integer attributes and 16 categorical attributes.

Preprocessing. Same as [21], the MNIST images undergo normalization, with all attributes scaled to the range of $[-1, 1]$. We adopt a different strategy for the tabular datasets from that used in [29]. Instead of transforming all attributes to binary form that can cause information loss and potential privacy breach [47], we convert categorical attributes into one-hot vectors while integer and continuous attributes are standardized using the following formula:

$$a = (a - \mu_a)/\sigma_a, \tag{22}$$

where μ_a and σ_a represent the mean and variance of the attribute a, respectively. The main architectures of generators for different datasets are similar but the last activation layers vary according to the different preprocessing. For generators trained on the MNIST dataset, we utilize the $\tanh(\cdot)$ function as the final activation layer. This choice aligns with the fact that all standardized attributes range from -1 to 1. In contrast, for generators trained on tabular datasets, the approach differs. Here, the identity function is applied as the final activation layer for normalized integer and continuous attributes, acknowledging that these do not have a fixed range. For one-hot attributes, we implement the Gumbel softmax function, as detailed in [28].

5.1.2 Baseline models. First of all, we employ WGAN_GP [21] as a baseline model, which is trained in a centralized manner, to establish a performance upper bound for models trained in a vertically partitioned manner. Next, we select VertiGAN and DPLT as comparative models to underscore the superiority of VFLGAN in vertically partitioned scenarios. Another baseline is a modified version of VFLGAN, named VFLGAN-base, which omits the second part of D_1 and D_2, allowing us to assess the impact of these modules. Readers can refer to Appendix A.5 for details of VFLGAN-base.

We apply WGAN_GP to evaluate the proposed Gaussian mechanism and choose GS-WGAN [7] and DPSGD GANs [57, 62] as baselines. This is due to their inability to provide a DP guarantee for our proposed VFLGAN. Notably, we maintain consistent generator and discriminator architectures across various differentially private mechanisms for a fair comparison. For comprehensive information on the architectures of the proposed VFLGAN and the baselines, please see **Appendix** A.4. Furthermore, we apply Proposition 4 to calculate the privacy budget for DPSGD GAN, ensuring a fair comparison by providing a tighter DP bound. This proposition is employed by both GS-WGAN and our mechanism for calculating the DP budget. We exclude PATE-GAN [30] in our baselines, as its conceptual framework is akin to GS-WGAN but presents challenges in training. Specifically, the authors of GS-WGAN [7] reported difficulties in training PATE-GAN on the MNIST dataset.

5.1.3 Utility Metrics. According to [29], there are two primary metrics for evaluating the utility of synthetic datasets: statistical similarity and AI training performance. For statistical similarity, we initially consider the Average Variant Distance (AVD) method, the same as [29, 50], which measures discrepancies between real and synthetic datasets. However, AVD is primarily applicable to discrete attributes, making it less suitable for our study. Instead, we utilize the Fréchet Distance (FD) [17], inspired by the Fréchet Inception Distance (FID) [23]. A lower FD indicates a closer resemblance between the real and synthetic datasets. We directly compute the FD using raw data for tabular datasets. However, individual image pixels lack meaningful semantic content, so for the MNIST dataset, we employ an Autoencoder to derive more semantically rich latent features, which are then used to calculate FD. The FD is calculated as,

$$FD(X, \tilde{X}) = \|\boldsymbol{\mu} - \tilde{\boldsymbol{\mu}}\|_2^2 + \mathrm{Tr}\left(\boldsymbol{V} + \tilde{\boldsymbol{V}} - 2\left(\boldsymbol{V}\tilde{\boldsymbol{V}}\right)^{1/2}\right), \tag{23}$$

where $\mathrm{Tr}(\cdot)$ calculates the trace of the input matrix, $\boldsymbol{\mu}$ and $\boldsymbol{V}$ denote the mean and variance of the real dataset X, and $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{V}}$ are those of

the synthetic dataset $\tilde{X}$. We evaluate AI training performance using accuracy and F1 scores in four different settings: training and testing on real data (TRTR), on synthetic data (TSTS), training on real data and testing on synthetic data (TRTS), and vice versa (TSTR). The TRTR setting serves as a baseline for AI model performance. The similarity of metrics in TSTS, TRTS, and TSTR settings to TRTR indicates how closely the synthetic data distribution resembles real data distribution. Notably, higher accuracy and F1 scores in TSTS, TRTS, and TSTR settings do not necessarily imply higher synthetic data quality. For the unlabeled synthetic MNIST data, we use the Inception Score (IS) [4], which assesses how convincing the synthetic images are in belonging to real labels. IS is calculated as:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\tilde{\boldsymbol{x}}} D_{KL}(p(y \mid \tilde{\boldsymbol{x}}) \| p(y))\right), \tag{24}$$

where $p(y \mid \tilde{\boldsymbol{x}})$) is the conditional class distribution from the Inception model [49], and $p(y) = \int_{\tilde{\boldsymbol{x}}} p(y \mid \tilde{\boldsymbol{x}}) p_g(\tilde{\boldsymbol{x}})$ is the marginal class distribution. The IS measures the distribution distance between $p(y)$ and $p(y \mid \tilde{\boldsymbol{x}})$. A higher IS indicates a more realistic generation of synthetic data. Since the Inception model is not optimized for MNIST, we replace it with an MLP model trained for digit classification on the MNIST dataset.

5.2 Utility Results of VFLGAN

In this section, we evaluate the performance of VFLGAN using the above-mentioned six datasets, chosen for their relevance and representativeness in diverse scenarios. We delve into a detailed comparison of VFLGAN against established benchmarks, showcasing its enhanced ability to capture intricate correlations among attributes distributed across different parties. Additionally, we employ a suite of robust evaluation metrics, carefully selected to provide a multi-faceted view of VFLGAN's performance. These metrics are designed to assess the distribution similarity between synthetic datasets and real datasets and the applicability of synthetic datasets in downstream tasks, thereby offering a holistic understanding of the model's capabilities and superiority in the realm of synthetic data generation.

5.2.1 MNIST Dataset. All models were trained for 300 epochs on the MNIST dataset, and the FD curves, as shown in the left part of Fig. 5, reveal insightful trends. WGAN_GP emerges as the top performer in terms of FD, with VFLGAN closely following. VFLGAN-base also surpasses VertiGAN, indicating a more effective generation of vertically partitioned data through VFL compared to HFL. For a practical demonstration, we selected models with the lowest FD across training epochs to generate synthetic images. The outcomes, as illustrated in Fig. 6, reveal a marked difference in the quality of the generated images. Specifically, the synthetic dataset generated by VFLGAN has an FD score approximately twice as high as the synthetic dataset produced by WGAN_GP, while VertiGAN's synthetic dataset has an FD score nearly seven times higher. This gap underscores the substantial advantage of VFLGAN over VertiGAN. Moreover, about half of VertiGAN's synthetic samples are either unrecognizable or lack continuity. On the other hand, while the VFLGAN-base produces more discernible images, some noise is evident, likely a result of information loss inherent to the VFL framework. To address this, we integrated an additional component into D_1 and D_2, i.e., the second part of D_1 and D_2 and corresponding loss functions, to enhance the similarity of the generated $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$ to the real samples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, as per Equations (14) and (15). This modification is evident in the improved clarity and recognizability of the synthetic data generated by VFLGAN. The right side of Fig. 5 illustrates the IS curves during training. The IS for real images sets a benchmark for synthetic ones. Consistent with the FD results, WGAN_GP leads in terms of generative model performance, with VFLGAN coming in third but maintaining a significant lead over VertiGAN. The IS performance of VFLGAN-base also exceeds that of VertiGAN. Ultimately, the alignment of performance rankings between IS and FD further affirms the superiority of VFLGAN in generating high-quality synthetic data.

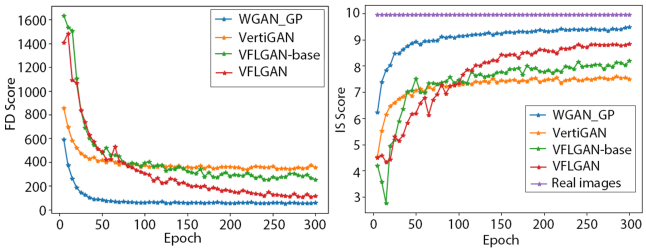


Figure 5: FD curves (lower is better) and IS curves (higher is better) on the MNIST dataset.

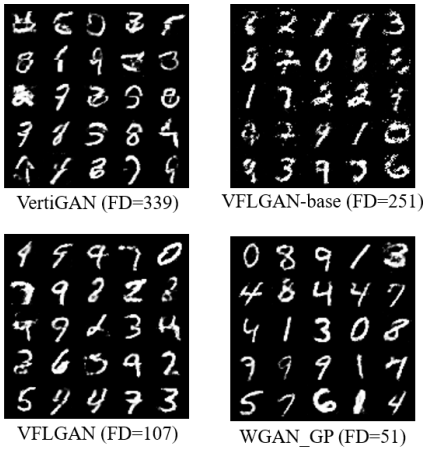


Figure 6: Synthetic digit samples.

5.2.2 Tabular Datasets. Now, we evaluate the proposed methods on five tabular datasets, i.e., Adult [5], Red-Wine-Quality and White-Wine-Quality [11], Credit [1], and HCV [42]. The upper five figures in Fig. 7 show the FD curves of various methods during training on the five datasets, and the lowest FD scores during training are highlighted in the figures. From Fig. 7, we can see that VFLGAN and VFLGAN-base show similar performance to WGAN_GP (trained in a centric manner) on all five datasets. Besides, VFLGAN and VFLGAN-base show superior performance than VertiGAN on all five datasets w.r.t. the lowest FD score, which means the performance of VFLGAN and VFLGAN-base almost reach the upper

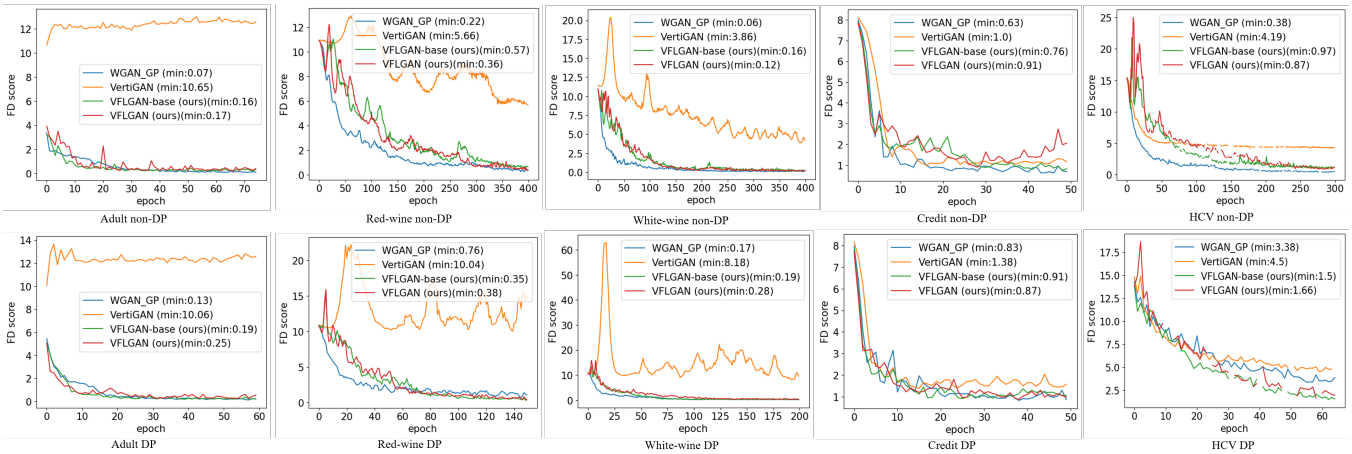


Figure 7: FD curves of different methods during training on the five datasets. The upper five figures show the FD curves of non-DP models. The lower five figures show the FD curves of DP models satisfying (10,δ)-DP. Minimum values on each curve are shown by legends. The discontinuity in the FD curves on HCV datasets is caused by nan value when calculating FD.

bound. Notably, the performance of VertiGAN w.r.t. FD score is far worse than other methods on Adult, Red-Wine-Quality, and White-Wine-Quality datasets while the performance of VertiGAN is close to other methods on Credit and HCV datasets. One possible reason is that some attributes have a large value range in Adult, Red-Wine-Quality, and White-Wine-Quality datasets and VertiGAN is not effective in learning the distribution of those attributes. On the contrary, the attributes' value ranges are limited in Credit and HCV datasets and VertiGAN achieves better performance w.r.t. FD.

We choose the models with the lowest FD among all training epochs to generate synthetic datasets with the same size as the real dataset. Then, we train random forest (RF) models to classify the data samples by the classification target attributes in the five datasets, using all other attributes under TRTR, TSTS, TRTS, and TSTR settings as introduced in Section 5.1.3. We apply the accuracy and F1 score to evaluate the performance of RF models. Note that under TRTR and TSTS settings, we apply cross-validation with 10 evenly split sub-sets and report the mean accuracy and F1 score. The experiment results of non-DP methods are shown in the upper five sub-tables in Table 1. TRTR setting provides the baseline accuracy and F1 score. The accuracy and F1 scores of other settings should be similar to those under the TRTR setting if the synthetic dataset is similar to the real dataset. Thus, we calculate the absolute difference of the accuracy and F1 scores between the TRTR and other settings as the evaluation metric. From the upper five sub-tables in Table 1, we can see that VFLGAN and VFLGAN-base generate better synthetic data than VertiGAN by a significant margin except for the HCV dataset. Moreover, the metrics of VFLGAN, VFLGAN-base, and WGAN_GP are very close, which means the performance of VFLGAN and VFLGAN-base almost reaches the upper bound. For the HCV dataset, the performance of the RF model is poor even for the TRTR setting, which means that this dataset is not suited for classification. In this case, we can refer to the FD for evaluation. As we mentioned before, VFLGAN, VFLGAN-base, and WGAN_GP outperform VertiGAN according to FD (Fig. 7).

Takeaway: The shared discriminator D_s plays a pivotal role in guiding the generators towards accurately learning the correlations among attributes distributed across different parties. Additionally, the second part of the discriminators D_1 and D_2 contributes to aligning the distribution of the synthetic data $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$ more closely with that of real data $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively. The performance gap between VFLGAN and VFLGAN-base is noticeable when the attribute number is large and correlations are closely related, e.g., MNIST, but the performance gap between VFLGAN and VFLGAN-base is ignorable when the number of attributes is small. Furthermore, our evaluation highlights the limitations of relying solely on TSTR metrics for assessing statistical utility, as evidenced by VertiGAN's comparable performance under TSTR on the Adult dataset. In some cases like HCV, some datasets are not suitable for classification, and we need to refer to FD for evaluation. This finding underlines the critical importance of utilizing a diverse set of evaluation methods, including TSTS, TRTS, and FD, to ensure a comprehensive analysis of synthetic datasets. Moreover, we observe that while k-way AVD applied in [29, 50] can measure the similarity of discrete attributes, it becomes computationally intensive as k increases and cannot support continuous data. Conversely, FD provides an efficient means to quantify the overall attribute similarity with minor computational demand, facilitating the selection of the optimal model with the lowest FD across training epochs. These insights advocate for the adoption of these metrics as standards in future research.

5.3 Utility Results of DP-VFLGAN

In this section, we first evaluate the effectiveness of the proposed Gaussian mechanism. Then, we evaluate the utility of the proposed DP-VFLGAN.

5.3.1 Effectiveness of the Proposed Gaussian Mechanism. We use the MNIST dataset to evaluate the effectiveness of different mechanisms. The DP budget is set as $(10, 1 \times 10^{-5})$, i.e., the generators trained by all mechanisms satisfy $(10, 1 \times 10^{-5})$-DP. As mentioned

Table 1: Classification Accuracy and F1 score of Random Forest Models to Evaluate the AI Training Utility of the Synthetic Data on the Wine-Quality Datasets and Adult Dataset. Absolute different metric values compared with the TRTR setting are shown in the bracket and the main evaluation metric is 'Total Difference', the lower the better.

	TRTR		TSTS		TRTS		TSTR		
Methods	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Total Difference**
Red-Wine-Quality									
WGAN_GP[21]	0.59	0.26	0.57 (0.02)	0.28 (0.02)	0.55 (0.03)	0.25 (0.01)	0.58 (0.01)	0.26 (0.00)	**0.09**
VertiGAN[29]	0.59	0.26	0.53 (0.06)	0.21 (0.05)	0.51 (0.08)	0.19 (0.07)	0.39 (0.20)	0.14 (0.12)	0.58
VFLGAN-base (ours)	0.59	0.26	0.55 (0.04)	0.27 (0.01)	0.51 (0.08)	0.21 (0.05)	0.55 (0.04)	0.27 (0.01)	0.23
VFLGAN (ours)	0.59	0.26	0.56 (0.03)	0.26 (0.00)	0.54 (0.05)	0.24 (0.02)	0.58 (0.01)	0.26 (0.00)	**0.11**
White-Wine-Quality									
WGAN_GP[21]	0.53	0.21	0.53 (0.00)	0.24 (0.03)	0.53 (0.00)	0.21 (0,00)	0.53 (0.00)	0.20 (0.01)	**0.04**
VertiGAN[29]	0.53	0.21	0.53 (0.00)	0.16 (0.05)	0.48 (0.05)	0.16 (0.05)	0.47 (0.06)	0.14 (0.07)	0.28
VFLGAN-base (ours)	0.53	0.21	0.53 (0.00)	0.21 (0.00)	0.52 (0.01)	0.20 (0.01)	0.52 (0.01)	0.20 (0.01)	**0.04**
VFLGAN (ours)	0.53	0.21	0.51 (0.02)	0.20 (0.01)	0.50 (0.03)	0.19 (0.02)	0.51 (0.02)	0.19 (0.02)	0.12
Adult Dataset									
WGAN_GP[21]	0.82	0.72	0.82 (0.00)	0.74 (0.02)	0.81 (0.01)	0.71 (0.01)	0.72 (0.10)	0.69 (0.03)	**0.17**
VertiGAN[29]	0.82	0.72	0.59 (0.23)	0.37 (0.35)	0.53 (0.29)	0.48 (0.24)	0.75 (0.07)	0.43 (0.29)	1.47
VFLGAN-base (ours)	0.82	0.72	0.81 (0.01)	0.66 (0.06)	0.80 (0.02)	0.68 (0.04)	0.76 (0.06)	0.72 (0.00)	**0.19**
VFLGAN (ours)	0.82	0.72	0.82 (0.00)	0.65 (0.07)	0.79 (0.03)	0.65 (0.07)	0.79 (0.03)	0.54 (0.18)	0.38
Credit Dataset									
WGAN_GP[21]	0.75	0.67	0.68 (0.07)	0.55 (0.12)	0.67 (0.08)	0.56 (0.11)	0.67 (0.08)	0.53 (0.14)	**0.60**
VertiGAN[29]	0.75	0.67	0.67 (0.08)	0.47 (0.20)	0.64 (0.11)	0.46 (0.21)	0.68 (0.07)	0.42 (0.25)	0.92
VFLGAN-base (ours)	0.75	0.67	0.72 (0.03)	0.67 (0.00)	0.64 (0.11)	0.51 (0.16)	0.63 (0.12)	0.58 (0.09)	**0.51**
VFLGAN (ours)	0.75	0.67	0.77 (0.02)	0.45 (0.22)	0.69 (0.06)	0.48 (0.19)	0.70 (0.05)	0.42 (0.25)	0.79
HCVEGY Dataset									
WGAN_GP[21]	0.23	0.20	0.29 (0.06)	0.17 (0.03)	0.23 (0.00)	0.21 (0.01)	0.24 (0.01)	0.16 (0.04)	**0.15**
VertiGAN[29]	0.23	0.20	0.30 (0.07)	0.21 (0.01)	0.25 (0.02)	0.22 (0.02)	0.24 (0.01)	0.14 (0.06)	**0.19**
VFLGAN-base (ours)	0.23	0.20	0.31 (0.08)	0.26 (0.06)	0.24 (0.01)	0.20 (0.00)	0.27 (0.04)	0.17 (0.03)	0.22
VFLGAN (ours)	0.23	0.20	0.29 (0.06)	0.22 (0.02)	0.28 (0.05)	0.24 (0.04)	0.25 (0.02)	0.17 (0.03)	0.22
Red-Wine-Quality $(10, 5 \times 10^{10-4})$**-DP**									
WGAN_GP[21]	0.59	0.26	0.56 (0.03)	0.27 (0.01)	0.56 (0.03)	0.25 (0.01)	0.56 (0.03)	0.24 (0.02)	**0.13**
VertiGAN[29]	0.59	0.26	0.62 (0.03)	0.13 (0.13)	0.46 (0.13)	0.16 (0.10)	0.40 (0.19)	0.10 (0.16)	0.74
DPLT	0.59	0.26	0.37 (0.22)	0.26 (0.00)	0.36 (0.23)	0.22 (0.04)	0.31 (0.28)	0.16 (0.10)	0.87
VFLGAN-base (ours)	0.59	0.26	0.59 (0.00)	0.23 (0.03)	0.59 (0.00)	0.23 (0.03)	0.57 (0.02)	0.21 (0.05)	**0.13**
VFLGAN (ours)	0.59	0.26	0.61 (0.02)	0.22 (0.04)	0.60 (0.01)	0.24 (0.02)	0.55 (0.04)	0.20 (0.06)	0.19
White-Wine-Quality $(10, 2 \times 10^{10-4})$**-DP**									
WGAN_GP[21]	0.53	0.21	0.55 (0.02)	0.24 (0.03)	0.53 (0.00)	0.20 (0.01)	0.52 (0.01)	0.20 (0.01)	**0.08**
VertiGAN[29]	0.53	0.21	0.70 (0.17)	0.22 (0.01)	0.48 (0.05)	0.17 (0.04)	0.38 (0.15)	0.12 (0.09)	0.51
DPLT	0.53	0.21	0.43 (0.10)	0.15 (0.06)	0.42 (0.11)	0.17 (0.04)	0.49 (0.04)	0.14 (0.07)	0.42
VFLGAN-base (ours)	0.53	0.21	0.54 (0.01)	0.19 (0.02)	0.53 (0.00)	0.19 (0.02)	0.52 (0.01)	0.17 (0.04)	**0.10**
VFLGAN (ours)	0.53	0.21	0.52 (0.01)	0.16 (0.05)	0.51 (0.02)	0.19 (0.02)	0.51 (0.02)	0.17 (0.04)	0.16
Adult Dataset $(10, 1 \times 10^{10-5})$**-DP**									
WGAN_GP[21]	0.82	0.72	0.84 (0.02)	0.78 (0.06)	0.82 (0.00)	0.72 (0.00)	0.75 (0.07)	0.71 (0.01)	**0.16**
VertiGAN[29]	0.82	0.72	0.66 (0.16)	0.40 (0.32)	0.56 (0.26)	0.50 (0.22)	0.75 (0.07)	0.43 (0.29)	1.32
DPLT	0.82	0.72	0.60 (0.22)	0.55 (0.17)	0.48 (0.34)	0.44 (0.28)	0.31 (0.51)	0.30 (0.42)	1.94
VFLGAN-base (ours)	0.82	0.72	0.82 (0.00)	0.72 (0.00)	0.80 (0.02)	0.69 (0.03)	0.82 (0.00)	0.72 (0.00)	**0.05**
VFLGAN (ours)	0.82	0.72	0.83 (0.01)	0.78 (0.06)	0.82 (0.00)	0.73 (0.01)	0.75 (0.07)	0.71 (0.01)	**0.16**
Credit Dataset $(10, 1 \times 10^{10-3})$**-DP**									
WGAN_GP[21]	0.75	0.67	0.78 (0.03)	0.70 (0.03)	0.74 (0.01)	0.63 (0.04)	0.71 (0.04)	0.63 (0.04)	**0.19**
VertiGAN[29]	0.75	0.67	0.70 (0.05)	0.44 (0.23)	0.64 (0.11)	0.48 (0.19)	0.69 (0.06)	0.42 (0.25)	0.89
DPLT	0.75	0.67	0.54 (0.21)	0.45 (0.22)	0.53 (0.22)	0.48 (0.19)	0.65 (0.10)	0.53 (0.14)	1.08
VFLGAN-base (ours)	0.75	0.67	0.66 (0.09)	0.43 (0.24)	0.63 (0.14)	0.49 (0.18)	0.69 (0.06)	0.45 (0.22)	0.93
VFLGAN (ours)	0.75	0.67	0.72 (0.03)	0.65 (0.02)	0.67 (0.08)	0.54 (0.13)	0.67 (0.08)	0.56 (0.11)	**0.45**
HCVEGY Dataset $(10, 8 \times 10^{10-4})$**-DP**									
WGAN_GP[21]	0.23	0.20	0.28 (0.05)	0.22 (0.02)	0.26 (0.03)	0.23 (0.03)	0.24 (0.01)	0.12 (0.08)	**0.22**
VertiGAN[29]	0.23	0.20	0.29 (0.06)	0.19 (0.01)	0.25 (0.02)	0.22 (0.02)	0.26 (0.03)	0.12 (0.08)	**0.22**
DPLT	0.23	0.20	0.41 (0.18)	0.17 (0.03)	0.30 (0.07)	0.25 (0.05)	0.25 (0.02)	0.11 (0.09)	0.44
VFLGAN-base (ours)	0.23	0.20	0.31 (0.08)	0.25 (0.05)	0.24 (0.01)	0.21 (0.01)	0.24 (0.01)	0.10 (0.10)	0.26
VFLGAN (ours)	0.23	0.20	0.31 (0.08)	0.20 (0.00)	0.23 (0.00)	0.21 (0.01)	0.25 (0.02)	0.17 (0.03)	**0.14**

TRTR: Train on real test on real; **TSTS**: train on synthetic test on synthetic; **TRTS**: train on real test on synthetic; **TSTR**: train on synthetic test on real; **Ac**: accuracy; **F1**: F1-score.

before, we apply the same network architecture as the official implementation of WGAN_GP when evaluating various differential privacy methods. The detailed architecture of WGAN_GP is presented in **Appendix** A.4. After training, we choose the generators with the lowest FD among training epochs to generate synthetic samples and the results are shown in Fig. 8. From the figure, we can see that our proposed Gaussian mechanism provides significantly better digits compared with that of DPSGD-GAN and GS_WGAN when satisfying the same DP guarantee. There are only two differences between our implementation of GS_GAN and that of the original paper. First, the model architectures are different for fair comparison. Second, we use unconditional GAN and conditional GAN is applied in the original paper. Considering the excellent results shown in [7], We think GS_WGAN relies on the advanced network architecture applied in the official implementation.

Takeaway: Compared with DPSGD, our proposed Gaussian mechanism introduces significantly less noise to the parameter gradients, thus preserving more useful information. Unlike GS_WGAN, our mechanism does not necessitate partitioning the training dataset into multiple subsets. It is widely recognized that smaller datasets

can adversely affect the performance of GANs. Consequently, when operating under the same privacy budget, the GAN trained with our Gaussian mechanism consistently outperforms its counterparts.

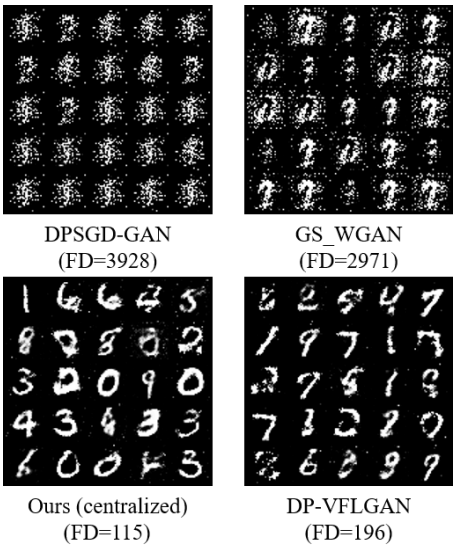


Figure 8: Synthetic digit samples satisfying $(10, 1 \times 10^{-5})$-DP.

5.3.2 Utility Results of DP-VFLGAN. We apply the proposed Gaussian mechanism to WGAN_GP to provide it with a DP guarantee. We also apply the proposed Gaussian mechanism to VertiGAN for fair comparison instead of using the DPSGD, which is applied in the original paper, since we have shown that our mechanism can provide a much tighter DP guarantee than DPSGD. We set $\epsilon = 10$ for all methods. For Adult and MNIST datasets, we set $\delta = 1 \times 10^{-5}$ like [7]. We set $\delta = 5 \times 10^{-4}$ and $\delta = 2 \times 10^{-4}$ for Red-Wine-Quality and White-Wine-Quality, respectively, since they are small datasets with only 1599 and 4898 records, respectively. We set $\delta = 8 \times 10^{-4}$ and $\delta = 1 \times 10^{-3}$ for HCV and Credit, respectively, since they are small datasets with only 1385 and 1000 records, respectively. We use the same T_{max} and σ for all the DL methods, which are computed according to Section 3.5 to meet the DP budget.

The performance of the proposed DP-VFLGAN on the MNIST dataset is shown in Fig. 8. We can see the FD of DP-VFLGAN is lower than the FDs of non-DP VFLGAN-base and non-DP VertiGAN in Fig 6, which shows the effectiveness of both VFLGAN and the proposed Gaussian mechanism. The FD curves during training are shown in the bottom five figures of Fig. 7. Similar to the non-DP methods, the DP-WGAN_GP, DP-VFLGAN, and DP-VFLGAN-base achieve lower FD than DP-VertiGAN. We do not show the FD performance of DPLT in Fig. 7 since it is too high compared with those of GAN-based methods. The AI training utility of different DP methods is shown in Table 1. We can see that compared with the non-DP versions of WGAN_GP, VFLGAN, and VFLGAN-base, the performance drop of the corresponding DP versions is limited, which shows the effectiveness of the proposed Gaussian mechanism. Moreover, the AI training utility of DP-VFLGAN and DP-VFLGAN-base is close to that of DP-WGAN_GP, which shows the effectiveness of the proposed VFLGAN framework. The performance increase of DP-VFLGAN-base and DP-VFLGAN on the Adult dataset may be caused by randomness during training. In addition, all GAN-based methods show better performance than DPLT w.r.t. the AI training utility.

5.4 Empirical Privacy Analysis

In this section, we first audit the privacy risk of the synthetic datasets generated by VFLGAN and DP-VFLGAN with ASSD. Then, we show that the proposed ASSD is robust compared to current MI attacks for synthetic datasets. Last, we estimate the privacy risk of intermediate features of ASIF.

Table 2: Mean and Standard Deviation of the MI attack Accuracy through Synthetic Datasets Generated by VFLGAN and DP-VFLGAN.

	Naive	Corr	Naive	Corr
	Record 1 (Adult)		**Record 2 (Adult)**	
Non-DP	0.65(0.09)	0.59(0.05)	0.58(0.10)	0.55(0.11)
$(10, 10^{-5})$-DP	0.46(0.06)	0.47(0.13)	0.51(0.12)	0.44(0.10)
$(5, 10^{-5})$-DP	0.50(0.09)	0.49(0.10)	0.50(0.08)	0.48(0.11)
	Record 1 (R wine)		**Record 2 (R wine)**	
Non-DP	0.59(0.11)	0.48(0.07)	0.51(0.05)	0.51(0.11)
$(10, 5 \times 10^{10-4})$-DP	0.53(0.12)	0.47(0.13)	0.52(0.18)	0.48(0.12)
$(5, 5 \times 10^{10-4})$-DP	0.48(0.11)	0.50(0.07)	0.43(0.09)	0.51(0.12)
	Record 1 (W wine)		**Record 2 (W wine)**	
Non-DP	0.55(0.12)	0.57(0.12)	0.60(0.06)	0.50(0.09)
$(10, 2 \times 10^{10-4})$-DP	0.54(0.11)	0.39(0.15)	0.52(0.05)	0.51(0.06)
$(5, 2 \times 10^{10-4})$-DP	0.44(0.03)	0.51(0.14)	0.53(0.09)	0.51(0.13)

Naive: The Adversary using naive feature; **Corr**: The Adversary using correlation feature; Non-DP: VFLGAN; (ϵ, δ)-DP: DP-VFLGAN providing (ϵ, δ)-DP guarantee.

5.4.1 Auditing Synthetic Dataset with ASSD. In implementing the MI Attack, our approach involved two distinct phases of generator training. First, we trained 100 generators on the entire private dataset, each with a unique random seed. Concurrently, we trained another set of 100 generators on the same dataset but excluding a specific target record, ensuring each generator had a different random seed. This setup enabled us to generate 200 synthetic datasets tailored to assess the presence of the target record. For the training of our model, we randomly selected 140 synthetic datasets—70 generated from the complete dataset and 70 excluding the target record. The remaining 60 datasets were used for testing purposes. The core of our analysis involved training a Random Forest model to identify whether the target record was part of the training data, based on the characteristics of these synthetic datasets. After testing the Random Forest model, we repeated the selection, training, and testing process five times to ensure robustness in our results. This repetition allowed us to calculate the mean and standard deviation of the attack's AUC score, providing a quantitative measure of privacy leakage. A higher AUC score indicated a more significant privacy risk, while an AUC score at or below 0.5 suggested minimal privacy exposure. Consistent with methodologies in previous studies [47], we employed naive and correlation features for the MI attacks. We decided not to include results from histogram features in our analysis, as they showed little potential for privacy compromise in our experiments.

We apply the methods of [47] and [38], to select the most vulnerable records in the training data with the proposed MI attack.

Table 3: AUC Performance of Different Attacking Methods on Synthetic Datasets.

Dataset \ Methods	LOGAN 0 [22]	LOGAN D1 [22]	MC [24]	GAN-leaks 0 [8]	GAN-leaks CAL [8]	DOMIAS [52]	ASSD(ours)
R-Wine (100/1000)	0.47	0.48	0.60	0.61	0.58	0.63	**0.98**
R-Wine (200/1000)	0.52	0.50	0.52	0.55	0.55	0.54	**0.95**
W-Wine (100/2000)	0.48	0.46	0.53	0.54	0.57	0.60	**0.87**
W-Wine (200/2000)	0.50	0.50	0.55	0.56	0.59	0.58	**0.92**
Housing (100/10000)	0.51	0.52	0.54	0.57	0.55	0.60	**0.81**
Housing (200/10000)	0.54	0.54	0.50	0.51	0.50	0.50	**0.89**

Dataset ($\#N_1/\#N_2$): the size of training dataset and test dataset is N_1 and the size of reference dataset is N_2. Note the proposed auditing method (ours) does not need test and reference datasets.

Detailed methodologies for selecting these vulnerable records are described in **Appendix** A.7. We chose the two most vulnerable records from each dataset for this analysis to measure the potential privacy leakage. The results of this assessment are presented in Table 2. We observe that the attack AUC scores on datasets generated by VFLGAN consistently exceed 50%, indicating a certain risk of privacy leakage. In contrast, datasets generated by DP-VFLGAN show attack AUC scores close to or below 50%. This suggests a substantial reduction in privacy risk, highlighting the effectiveness of the DP-VFLGAN in protecting against MI attacks.

5.4.2 Evaluating the Effectiveness of ASSD. Now, we compare the proposed ASSD with several current MI attacks for synthetic datasets proposed in [8, 22, 24, 52] on three datasets, i.e., Red-Wine-Quality, White-Wine-Quality, and Housing [43]. Housing is applied in the benchmark proposed in [52]. Since those MI attacks cannot support categorical attributes, we add two new datasets to the benchmark with minimum categorical attributes, i.e., Red-Wine-Quality and White-Wine-Quality. We also apply We set the size of training datasets to 100 and 200. All the attacks in [8, 22, 24, 52] require a test dataset with the same size as the training dataset. Moreover, Domias, Logan D1, and Gan-leak cal require a reference dataset, which presents the same distribution as the training dataset. On the contrary, the proposed ASSD does not require a test or reference dataset. The attack AUC scores are shown in Table 3. ASSD present a superior performance than other attacks, which shows the effectiveness of the proposed auditing scheme. We can infer that current attacks in [8, 22, 24, 52] won't succeed when the AUC score of ASSD is small (e.g., less than ~0.6).

Takeaway: ASSD is more effective than other attacks because ASSD targets the vulnerable record in the training dataset selected by the method of [47] while other attacks target every record in the training dataset. Besides, ASSD applies the leave-one-out setting, which is stronger than auxiliary datasets but more realistic and practical for data holders. The notable disparity in AUC scores of ASSD between Table 3 and Table 2 can be attributed to variations in the sizes of the respective training datasets. In Table 3, the size of training datasets is 100 and 200 while in Table 2, the whole datasets are used to train the models. It's well known that a smaller size can cause more privacy leakage [52].

5.4.3 Auditing Intermediate Features with ASIF. The implementation of ASIF is similar to that of ASSD in Section 5.4.1. We also train 200 pairs (D_1 and D_2) of local discriminators, in which 100 pairs are trained on the complete dataset and 100 pairs are trained on the same dataset but excluding a specific target record (x_t), which is selected by the methods in [47]. We input the complete dataset to the local discriminators to generate 200 sets of intermediate features, of which 140 sets are used for training, and 60 sets are used for the test. Then, we train a random forest model to detect whether x_t is in the training dataset given a set of intermediate features. From Table 4, we can see that the privacy risk of intermediate features is minor, even for non-DP models.

Table 4: Mean and Standard Deviation of the MI attack Accuracy through Intermediate Features Generated by VFLGAN and DP-VFLGAN.

	Naive	Corr	Naive	Corr
	Record 1 (Adult)		**Record 2 (Adult)**	
Non-DP	0.49(0.12)	0.58(0.09)	0.53(0.09)	0.55(0.12)
$(10, 10^{-5})$-DP	0.49(0.14)	0.50(0.08)	0.54(0.11)	0.48(0.12)
$(5, 10^{-5})$-DP	0.49(0.03)	0.50(0.12)	0.54(0.09)	0.51(0.10)
	Record 1 (R wine)		**Record 2 (R wine)**	
Non-DP	0.54(0.09)	0.55(0.14)	0.52(0.17)	0.54(0.12)
$(10, 5\times10^{10-4})$-DP	0.53(0.04)	0.50(0.07)	0.54(0.09)	0.50(0.09)
$(5, 5\times10^{10-4})$-DP	0.51(0.09)	0.45(0.03)	0.54(0.02)	0.53(0.07)
	Record 1 (W wine)		**Record 2 (W wine)**	
Non-DP	0.52(0.10)	0.50(0.08)	0.51(0.09)	0.49(0.13)
$(10, 2\times10^{10-4})$-DP	0.52(0.10)	0.52(0.12)	0.48(0.10)	0.51(0.12)
$(5, 2\times10^{10-4})$-DP	0.49(0.14)	0.56(0.07)	0.52(0.07)	0.53(0.08)

Naive: The Adversary using naive feature; **Corr**: The Adversary using correlation feature; Non-DP: VFLGAN; (ϵ, δ)-DP: DP-VFLGAN providing (ϵ, δ)-DP guarantee.

6 CONCLUSION AND FUTURE WORK

In this paper, we found that VertiGAN published in PETS 2023 [29] can not effectively learn the correlation among attributes between different parties, which leads to the distribution of synthetic data being different from that of the real data. To resolve this issue, we proposed the *first* VFL-based GAN, VFLGAN, for vertically partitioned data publication. Experiment results show that VFLGAN significantly improves the quality of synthetic data compared with the state-of-the-art methods. We also proposed a Gaussian mechanism for VFLGAN to make the generators satisfy a rigorous DP guarantee. Experimental results show that the proposed Gaussian mechanism can produce better synthetic data compared with current differentially private mechanisms under the same DP budget. Besides, the utility drop of DP-VFLGAN is limited compared to its non-DP version. Additionally, we propose a practical privacy leakage measurement with realistic assumptions since the DP is too conservative. Our experimental results show that DP-VFLGAN can effectively mitigate privacy breaches. However, the proposed privacy leakage measurement can only estimate the privacy breach from the view of external attackers. Our future work will be on estimating the privacy breach from the view of internal attackers.

ACKNOWLEDGMENTS

This work was supported in part by the Asian Institute of Digital Finance (AIDF) under grant A-0003504-09-00.

REFERENCES

[1] 2019. South German Credit. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5X89F.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.

[3] Borja Balle, Giovanni Cherubin, and Jamie Hayes. 2022. Reconstructing Training Data with Informed Adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*. 1138–1156. https://doi.org/10.1109/SP46214.2022.9833677

[4] Shane Barratt and Rishi Sharma. 2018. A note on the inception score. *arXiv preprint arXiv:1801.01973* (2018).

[5] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[6] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, 35.

[7] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. 2020. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. *Advances in Neural Information Processing Systems* 33 (2020), 12673–12684.

[8] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 343–362.

[9] Hao Chen, Kim Laine, and Peter Rindal. 2017. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1243–1255.

[10] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 670–680. https://doi.org/10.18653/v1/D17-1070

[11] Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. 2009. Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[13] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems* 29 (2016).

[14] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[15] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer, 486–503.

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

[17] Maurice Fréchet. 1957. Sur la distance de deux lois de probabilité. In *Annales de l'ISUP*, Vol. 6. 183–198.

[18] Matteo Giomi, Franziska Boenisch, Christoph Wehmeyer, and Borbála Tasnádi. 2023. A Unified Framework for Quantifying Privacy Risk in Synthetic Data. In *Proceedings on Privacy Enhancing Technologies*.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[20] Florent Guépin, Matthieu Meeus, Ana-Maria Cretu, and Yves-Alexandre de Montjoye. 2023. Synthetic is all you need: removing the auxiliary data assumption for membership inference attacks against synthetic data. *arXiv preprint arXiv:2307.01701* (2023).

[21] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. *Advances in neural information processing systems* 30 (2017).

[22] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies* (2019).

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

[24] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. 2019. Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models. *Proceedings on Privacy Enhancing Technologies* (2019).

[25] F Houssiau, J Jordon, SN Cohen, O Daniel, A Elliott, J Geddes, C Mole, C Rangel-Smith, and L Szpruch. 2022. TAPAS: a toolbox for adversarial privacy auditing of synthetic data. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*. Neural Information Processing Systems Foundation.

[26] Yan Huang, David Evans, and Jonathan Katz. 2012. Private set intersection: Are garbled circuits better than custom protocols?. In *NDSS*.

[27] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. 2020. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems* 33 (2020), 22205–22216.

[28] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[29] Xue Jiang, Yufei Zhang, Xuebing Zhou, and Jens Grossklags. 2023. Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data. In *Proceedings on Privacy Enhancing Technologies*.

[30] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*.

[31] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[32] Galen Andrew Krishna Pillutla, Peter Kairouz, H Brendan McMahan, Alina Oprea, and Sewoong Oh. 2023. Unleashing the power of randomization in auditing differentially private ml. *Advances in Neural Information Processing Systems* (2023).

[33] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. 2022. Vertical federated learning. *arXiv preprint arXiv:2211.12814* (2022).

[34] Fred Lu, Joseph Munoz, Maya Fuchs, Tyler LeBlond, Elliott Zaresky-Williams, Edward Raff, Francis Ferraro, and Brian Testa. 2022. A general framework for auditing differentially private machine learning. *Advances in Neural Information Processing Systems* 35 (2022), 4165–4176.

[35] Pei-Hsuan Lu, Pang-Chieh Wang, and Chia-Mu Yu. 2019. Empirical evaluation on synthetic data generation with generative adversarial network. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics*. 1–6.

[36] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. *Advances in neural information processing systems* 32 (2019).

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[38] Matthieu Meeus, Florent Guepin, Ana-Maria Cretu, and Yves-Alexandre de Montjoye. 2023. Achilles' Heels: Vulnerable Record Identification in Synthetic Data Publishing. *arXiv preprint arXiv:2306.10308* (2023).

[39] Luise Mehner, Saskia Nuñez von Voigt, and Florian Tschorsch. 2021. Towards explaining epsilon: A worst-case study of differential privacy risks. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 328–331.

[40] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[41] Noman Mohammed, Dima Alhadidi, Benjamin C.M. Fung, and Mourad Debbabi. 2014. Secure Two-Party Differentially Private Data Release for Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 11, 1 (2014), 59–71. https://doi.org/10.1109/TDSC.2013.22

[42] Mahmoud Nasr, Khaled El-Bahnasy, M. Hamdy, and Sanaa M. Kamal. 2019. Hepatitis C Virus (HCV) for Egyptian patients. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5989V.

[43] R Kelley Pace and Ronald Barry. 1997. Sparse spatial autoregressions. *Statistics & Probability Letters* 33, 3 (1997), 291–297.

[44] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Ulfar Erlingsson. 2018. Scalable Private Learning with PATE. In *International Conference on Learning Representations*.

[45] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.

[46] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. 3–18.

[47] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic Data - Anonymisation Groundhog Day. In *USENIX Security Symposium*.

[48] Thomas Steinke, Milad Nasr, and Matthew Jagielski. 2024. Privacy auditing with one (1) training run. *Advances in Neural Information Processing Systems* 36 (2024).

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[50] Peng Tang, Xiang Cheng, Sen Su, Rui Chen, and Huaxi Shao. 2021. Differentially Private Publication of Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 18, 2 (2021), 780–795. https://doi.org/10.1109/TDSC.2019.2905237
[51] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2* (2019).
[52] Boris van Breugel, Hao Sun, Zhaozhi Qian, and Mihaela van der Schaar. 2023. Membership inference attacks against synthetic data through overfitting detection. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
[53] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.
[54] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1226–1235.
[55] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. 2018. Improving the improved training of wasserstein gans: A consistency term and its dual effect. *ICLR*.
[56] Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. 2018. Wasserstein divergence for gans. In *Proceedings of the European conference on computer vision (ECCV)*. 653–668.
[57] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739* (2018).
[58] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P Bennett. 2019. Assessing privacy and quality of synthetic health data. In *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse*. 1–4.
[59] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. 2019. Neural Network Inversion in Adversarial Setting via Background Knowledge Alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 225–240.
[60] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. 2022. Enhanced Membership Inference Attacks against Machine Learning Models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 3093–3106.
[61] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially private model publishing for deep learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 332–349.
[62] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594* (2018).

A APPENDIX

A.1 Literature Review

This section presents a detailed literature review. In this regard, we first introduce the recent works about vertically partitioned data publication methods that provide DP guarantees. Next, we review multiple differentially private mechanisms for GANs and explain why they are unsuitable for the proposed VFLGAN. Last, we describe some privacy measurements and attacks on synthetic datasets inspired by which we propose our practical auditing schemes to measure privacy leakage.

A.1.1 Vertically Partitioned Data Publication. In [41], the authors proposed a secure two-party algorithm, DistDiffGen, that applies the exponential mechanism for vertically partitioned data publication and satisfies ϵ-DP. However, their protocol only works under the two-party scenario and the data utility deteriorates fast when the number of attributes increases [41]. Besides, DistDiffGen is tailored to classification tasks and cannot guarantee meaningful utility for many common data analysis tasks [50]. In [50], the authors proposed DPLT for vertically partitioned data publication, which satisfies ϵ-DP. However, DPLT is limited to discrete datasets. Besides, DPLT evenly splits the privacy budget over all the attribute pairs, which is unreasonable since the information leakage levels of different attributes are usually different. As pointed out in [29], the noise scale may increase exponentially with the increased data dimensionality, which can cause a significant utility loss. The first GAN-based vertically partitioned data publication method, VertiGAN, was proposed in [29] to solve the above issues. VertiGAN satisfies (ϵ, δ)-DP, where the DP guarantee is achieved by adding noise when updating discriminators' parameters so the privacy budget can be distributed more intelligently among attributes and the curse of dimensionality can be mitigated. However, the discriminator is updated according to FedAvg [37], a Horizontal Federated Learning (HFL)-based method. However, VeriGAN is less effective in learning the correlation among the attributes of different parties. This paper applies a VFL framework to learn the correlation mentioned above.

A.1.2 Differentially Private Generative Adversary Networks. For general deep learning models, there are mainly two kinds of training methods to make the trained models satisfy differential privacy, i.e., Differentially Private Stochastic Gradient Descent (DPSGD) [45, 61] and Private Aggregation of Teacher Ensembles (PATE) [44]. In [57, 62], the authors proposed variants of DPSGD to train the discriminator and apply non-private SGD to train the generator. The rationale is that only the discriminator can access the training data, and the generator learns from the discriminator to generate better synthetic data. According to the post-processing theorem, Proposition 5, the generator satisfies the same level of DP as the discriminator [57, 62]. However, this method is unsuitable for our VFLGAN, shown in Fig. 4, since each party cannot control the parameter update of the shared discriminator. PATE-GAN [30] adopts the PATE framework to provide differential privacy. First, multiple non-private discriminators are trained on non-overlapping subsets. Then, the non-private discriminators are applied to train a student discriminator that satisfies (ϵ, δ)-DP. Last, the student discriminator is used to train the generator. However, this method is inapplicable to our VFL-based scheme since the non-private teacher discriminators can cause information leakage through the shared discriminator. GS-WGAN [7] provides another solution. The discriminators are trained in a non-private way while the backward gradients between the discriminators and the generator are sanitized with a Gaussian mechanism to make the generator satisfy a (ϵ, δ)-DP. Besides, the privacy guarantee is enhanced by the subsampling procedure [54], i.e., dividing the training dataset into multiple non-overlapping subsets as [30] and training multiple discriminators on each subset. However, this framework cannot be adapted to VFLGAN since the information can be leaked through the shared discriminator when training non-private discriminators.

A.1.3 Privacy Measurements and Attacks on Synthetic Dataset. Some papers [18, 35, 58] utilize the distribution similarity between the synthetic dataset and the training dataset to measure the potential privacy leakage. In [58], the privacy loss is measured by nearest neighbour adversarial accuracy, and both training and test datasets are required to calculate the privacy loss. In [35], the authors first sort the original and synthetic data records according to a predefined metric, then compare the number of matches according to the rank of ordered records where more matches indicate a higher risk of re-identification. In [18], the authors proposed singling out

attack, linkability attack, and inference attack against synthetic datasets, which assume there is a control dataset following the same distribution as the training dataset and measured the privacy leakage by the attack accuracy. However, the above works do not follow the DP principle when measuring privacy leakage. Instead, they mainly measure the overfitting level of the GAN to the training data as privacy leakage.

The principle of DP was first adopted in [47] to design privacy attacks against synthetic datasets. The authors of [47] apply shadow models [46] to launch an MI attack that tries to distinguish whether the training dataset contains a specific record given the generated synthetic dataset. Shadow models MI attack is also applied in [25], which is further enhanced by an advanced feature map. However, these shadow model attacks require a large amount of reference data which follows the same distribution as the training dataset. For example, the size of the reference dataset is ten times larger than the training dataset in [47]. The feasibility of this assumption in the context of privacy auditing is questionable, as expecting the data publisher to employ an excessively large dataset for auditing purposes is impractical. Moreover, as pointed out in [60], improper reference datasets can overestimate or underestimate the privacy loss. In [20], the authors proposed an MI attack requiring only the targeted synthetic dataset. However, the double-counting phenomenon has not been resolved. In this paper, we adopt the leave-one-out setting proposed in [60] and shadow models to measure the privacy leakage of any specific record, which satisfies the principle of DP and does not require a reference dataset.

There is another research line about MI attacks on synthetic datasets [8, 22, 24, 52]. These attacks have the same black-box assumption as our auditing method, i.e., the GAN models are not accessible to attackers. However, the attacks in [8, 22, 24, 52] require auxiliary datasets and targets every record in the training dataset. On the contrary, our auditing method does not require any auxiliary dataset so that data holders can use all of their data to train the generative model. Besides, our auditing method targets one vulnerable data record and thus achieves superior performance than the attacks in [8, 22, 24, 52]. On the other hand, [27, 32, 34, 48] aim to estimate the lower bounds on ϵ for ϵ-DP. There are three major differences between our auditing method and those in [27, 32, 34, 48]. First of all, methods in [27, 32, 34, 48] estimate the worst privacy risk of ALL possible inputs while our auditing method estimates the privacy risk of the record in training datasets (which is also the focus of data holders). Second, methods in [27, 32, 34, 48] are designed for classification models and apply the classification losses to conduct attacks. However, there is no such classification loss in our VFLGAN. Third, in the vertically partitioned data publication scenario, models are assumed to be not accessible to attackers, as in [8, 22, 24, 52]. However, methods in [27, 32, 34, 48] require submitting requests to the classification models and receiving the results that are used to launch the attacks.

A.2 Proof of Theorem 1

Proof. Let $\boldsymbol{x}_i^B$ and $\boldsymbol{x}_i^{B'} \in X_i$ denote two adjacent mini-batches of training data of party i. The gradients of the parameters of the first layer of D_i are clipped using (20). Then the L2 norm of those gradients has the following upper bound,

$$\left\| clip(\mathcal{G}^1_{D_i(\boldsymbol{x}_j^B)}, C) \right\|_2 \leq C, \quad i, j \in \{1, 2\}. \tag{25}$$

Note that although the input of D_i is $\boldsymbol{x}_i^B$, the gradients of D_i are affected by both $\boldsymbol{x}_1^B$ and $\boldsymbol{x}_2^B$ according to (14). According to the triangle inequality, the L2 sensitivity of the parameters can be derived as

$$\Delta_2 f = \max_{\boldsymbol{x}_j^B, \boldsymbol{x}_j^{B'}} \left\| clip(\mathcal{G}^1_{D_i(\boldsymbol{x}_j^B)}, C) - clip(\mathcal{G}^1_{D_i(\boldsymbol{x}_j^{B'})}, C) \right\|_2 \leq 2C, \tag{26}$$

where $i, j \in \{1, 2\}$. According to Proposition 1, $\mathcal{G}^1_{D_i}$ computed by (21) satisfies $(\alpha, \alpha/(2\sigma^2))$-RDP w.r.t. $\boldsymbol{x}_1^B$ and $\boldsymbol{x}_2^B$ since original gradients are calculated regarding $\boldsymbol{x}_1^B$ and $\boldsymbol{x}_2^B$. Let $\boldsymbol{x}^B = [\boldsymbol{x}_1^B, \boldsymbol{x}_2^B]$ and $\boldsymbol{x}^{B'} = [\boldsymbol{x}_1^{B'}, \boldsymbol{x}_2^{B'}]$ be two adjacent mini-batches collected from the complete dataset, i.e., $X = [X_1, X_2]$. Equations (25) and (26) still hold if we delete the subscript of $\boldsymbol{x}_j^B$ and $\boldsymbol{x}_j^{B'}$. Thus, similar to the above proving process, $\mathcal{G}^1_{D_i}$ computed by (21) also satisfies $(\alpha, \alpha/(2\sigma^2))$-RDP w.r.t. $\boldsymbol{x}^B$.

According to Proposition 5 (Post-processing theorem), the parameters of the first layer of D_1 and D_2 updated by

$$\boldsymbol{\theta}^1_{D_i} = \boldsymbol{\theta}^1_{D_i} - \eta_{D_i} \mathcal{G}^1_{D_i} \tag{27}$$

satisfy the same RDP as $\mathcal{G}^1_{D_i}$.

The outputs of the first part of D_1, D_2, i.e., the intermediate features $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, can be expressed as,

$$\boldsymbol{f}_i = func_{D_i}(\boldsymbol{\theta}^1_{D_i} \boldsymbol{x}_i), \quad i \in \{1, 2\}, \tag{28}$$

where $\boldsymbol{x}_i$ denotes the input of D_i, $func_{D_i}$ denotes the calculation after the first layer ($\boldsymbol{\theta}^1_{D_i} \boldsymbol{x}_i$). Thus, according to Proposition 5, since $\boldsymbol{\theta}^1_{D_i}$ satisfies $(\alpha, \alpha/(2\sigma^2))$-RDP, the mechanism $func_{D_i}(\boldsymbol{\theta}^1_{D_i} \boldsymbol{x}_i)$ in (28) satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP, i.e., the first part of D_1 and D_2 satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP.

On the other hand, according to (17), the generators G_1 and G_2 can only learn the information about the input data from the gradients of the first layer of D_1 and D_2, respectively. During the back-propagation process, let $\boldsymbol{\delta}^G_{D_i}$ and $\boldsymbol{\delta}^1_{D_i}$ denote the backward gradients before and after the first layer of D_i, which is illustrated in Fig. 9. Note that the parameters of the first layer of D_i are updated according to $\boldsymbol{\delta}^1_{D_i}$ and the parameters of G_i are updated according to $\boldsymbol{\delta}^G_{D_i}$. The relationship between $\boldsymbol{\delta}^G_{D_i}$ and $\boldsymbol{\delta}^1_{D_i}$ can be expressed as

$$\boldsymbol{\delta}^G_{D_i} = \boldsymbol{\delta}^1_{D_i} \boldsymbol{\theta}^1_{D_i}. \tag{29}$$

According to Proposition 5, $\boldsymbol{\delta}^G_{D_i}$ satisfies the same RDP as $\boldsymbol{\theta}^1_{D_i}$. Since the parameters of G_i is updated according to $\boldsymbol{\delta}^G_{D_i}$, G_i satisfies $(\alpha, \alpha/(2\sigma^2))$-RDP w.r.t. x_1^B, x_2^B, and x^B. □

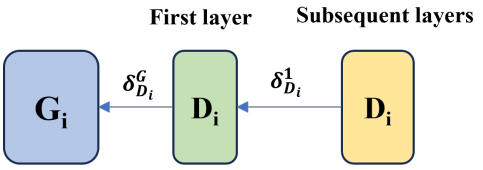
A.3 Auditing Scheme for Intermediate Features

The following privacy game can define the MI attack applied in ASIF. **(i)** First, the challenger selects a fixed target record $\boldsymbol{x}_t$ from a given training dataset X. **(ii)** Then, the challenger trains a VFLGAN $\boldsymbol{\theta}_0 \xleftarrow{s_0} \mathcal{T}(X \setminus \boldsymbol{x}_t)$ on $X \setminus \boldsymbol{x}_t$ (dataset X excluding $\boldsymbol{x}_t$) by using a fresh random seed s_0 in the training algorithm $\mathcal{T}$. In this paper, $\mathcal{T}$

Table 5: Classification Accuracy and F1 score of Random Forest Models to Evaluate the AI Training Utility of the Synthetic Data on the Wine-Quality Datasets. Absolute different metric values compared with the TRTR setting are shown in the bracket, and the main evaluation metric is 'Total Difference'; lower is better.

Normalized Integer Representation for the Quality attribute (red wine)									
	TRTR		TSTS		TRTS		TSTR		Total Difference
Models	Acc	F1	Acc	F1	Acc	F1	Acc	F1	
WGAN_GP[21]	0.59	0.26	0.62 (0.03)	0.45 (0.19)	0.53 (0.06)	0.20 (0.06)	0.54 (0.05)	0.29 (0.03)	0.42
VertiGAN[29]	0.59	0.26	0.92 (0.33)	0.67 (0.41)	0.80 (0.21)	0.41 (0.15)	0.55 (0.04)	0.26 (0.00)	1.14
VFLGAN-base (ours)	0.59	0.26	0.74 (0.15)	0.28 (0.02)	0.66 (0.07)	0.25 (0.01)	0.56 (0.03)	0.26 (0.00)	0.28
VFLGAN (ours)	0.59	0.26	0.74 (0.15)	0.44 (0.18)	0.59 (0.00)	0.29 (0.03)	0.51 (0.08)	0.26 (0.00)	0.44
Normalized Integer Representation for the Quality attribute (white wine)									
WGAN_GP[21]	0.53	0.21	0.49 (0.04)	0.17 (0.04)	0.48 (0.05)	0.14 (0.07)	0.53 (0.00)	0.19 (0.02)	0.22
VertiGAN[29]	0.53	0.21	0.77 (0.24)	0.48 (0.27)	0.46 (0.07)	0.15 (0.06)	0.38 (0.15)	0.14 (0.07)	0.86
VFLGAN-base (ours)	0.53	0.21	0.52 (0.01)	0.28 (0.07)	0.46 (0.07)	0.20 (0.01)	0.47 (0.06)	0.19 (0.02)	0.24
VFLGAN (ours)	0.53	0.21	0.50 (0.03)	0.24 (0.03)	0.47 (0.06)	0.18 (0.03)	0.50 (0.03)	0.21 (0.00)	0.18

TRTR: Train on real test on real; **TSTS**: train on synthetic test on synthetic; **TRTS**: train on real test on synthetic; **TSTR**: train on synthetic test on real; **Ac**: accuracy; **F1**: F1-score.



D: Discriminator;
G: Generator;
$\delta^1_{D_i}$: Gradients backwards to the first linear layer of D_i ;
$\delta^G_{D_i}$: Gradients backwards from the first linear layer of D_i .

Figure 9: Backward loss before and after the first layer of the discriminators.

refers to Algorithm 1. **(iii)** The challenger trains another VFLGAN $\theta_1 \xleftarrow{s_1} \mathcal{T}(X)$ on X by using a fresh random seed s_1 in algorithm $\mathcal{T}$. **(iv)** The challenger inputs the whole dataset X into $\theta_0(\cdot)$ and $\theta_1(\cdot)$ to generate two sets of intermediate features f_0 and f_1, respectively. **(v)** The challenger flips a random unbiased coin $b \in \{0, 1\}$ and sends the set of intermediate features and target record, f_b and x_t, to the adversary. **(vi)** The adversary tries to figure out the true b based on the observation of $\{f_b, x_t\}$, i.e., $\hat{b} \leftarrow \mathcal{A}(f_b, x_t)$. **(vii)** Last, if $\hat{b} = b$, the adversary wins. Otherwise, the challenger wins. This MI attack is summarized in Algorithm 4.

Now we describe how to train the adversary model $\mathcal{A}$ in Algorithm 4. We apply the shadow modelling approach [46] to train the adversary model $\mathcal{A}$ through the following process. **(i)** First, train M VFLGANs, $\theta_{0_{1:M}}$, on dataset $X \setminus x_t$ by using different random seeds $s_{0_{1:M}}$ in the training algorithm $\mathcal{T}$. **(ii)** Apply $\theta_{0_{1:M}}$ to generate M intermediate features $f_{0_{1:M}}$. **(iii)** Train M VFLGANs, $\theta_{1_{1:M}}$, on dataset X by using different random seeds $s_{1_{1:M}}$ in the training algorithm $\mathcal{T}$. **(vi)** Apply $\theta_{1_{1:M}}$ to generate M sets of intermediate features $f_{1_{1:M}}$. **(v)** Extract features of the intermediate features. **(vi)** Train the adversary model $\mathcal{A}$ to distinguish whether the features are from $f_{0_{1:M}}$ or $f_{1_{1:M}}$. The training process is summarized in Algorithm 5.

Algorithm 4: Membership Inference Attack

Input: Training algorithm $\mathcal{T}$; dataset X; target record x_t; unbiased coin b; fresh random seeds s_0 and s_1; VFLGAN θ_i; intermediate feature f.
Output: Success or failure.

$\theta_0 \xleftarrow{s_0} \mathcal{T}(X \setminus x_t)$
$\theta_1 \xleftarrow{s_1} \mathcal{T}(X)$
$f_0 \longleftarrow \theta_0(X)$ & $f_1 \longleftarrow \theta_1(X)$
$b \sim \{0, 1\}$
$\hat{b} \longleftarrow \mathcal{A}(f_b, x_t)$
if $\hat{b} == b$ **then**
  Output success
else
  Output failure
end if

Algorithm 5: Training the Adversary of MI Attack

Input: Training algorithms $\mathcal{T}$ and $\mathcal{T}_{\mathcal{A}}$; dataset X; target record x_t; random seeds $s_{0_{1:M}}$ and $s_{1_{1:M}}$; feature extraction function $Extr(\cdot)$; intermediate feature f.
Output: Trained $\mathcal{A}$.

$\theta_{0_{1:M}} \xleftarrow{s_{0_{1:M}}} \mathcal{T}(X \setminus x_t)$
$f_{0_{1:M}} \longleftarrow \theta_{0_{1:M}}(X)$
$\theta_{1_{1:M}} \xleftarrow{s_{1_{1:M}}} \mathcal{T}(X)$
$f_{1_{1:M}} \longleftarrow \theta_{1_{1:M}}(X)$
$Feat_{0_{1:M}} \longleftarrow Extr(f_{0_{1:M}})$ & $Feat_{1_{1:M}} \longleftarrow Extr(f_{1_{1:M}})$
$\mathcal{A} \longleftarrow \mathcal{T}_{\mathcal{A}}(Feat_{0_{1:M}}, Feat_{1_{1:M}})$

A.4 Model Architectures

Figure 10 shows the architecture of WGAN_GP, which is applied when we evaluate the effectiveness of various differentially private mechanisms in Section 5.3.1. Our implementation of VertiGAN (no public official code) also applies the same architecture as shown in Fig. 10. In the proposed VFLGAN, the architecture of the generator is the same as that of the generator in Fig. 10 while the architecture of discriminators is shown in Fig. 11.

A.5 Details of VFLGAN-base

Figure 12 shows the frameworks of VFLGAN-base. To construct VFLGAN-base, we delete the second part (shown with a deeper

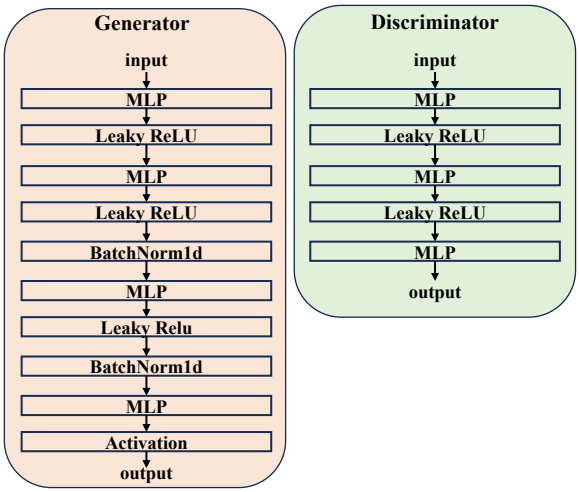


Figure 10: Architectures of WGAN_GP.

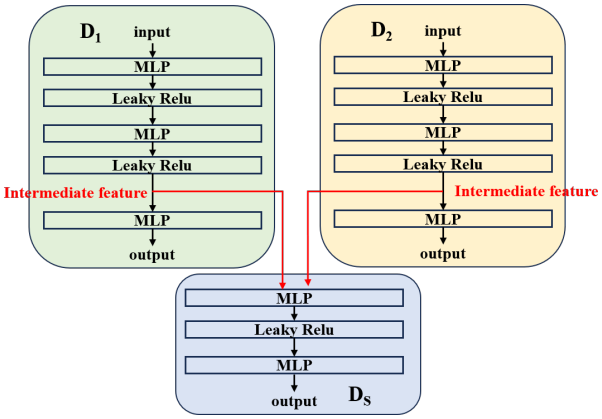


Figure 11: Architectures of the discriminators of VFLGAN.

colour) of D_1 and D_2 of VFLGAN (shown in Fig. 4) and their loss functions (L_{D_1} and L_{D_2}). L_{D_i}, $i \in \{1, 2\}$, can push the distribution of $\tilde{\boldsymbol{x}}_i$ to be similar to that of $\boldsymbol{x}_i$. L_{D_s} can push the distribution of $[\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2]$ to be similar to that of $[\boldsymbol{x}_1, \boldsymbol{x}_2]$. VFLGAN-base serves as an ablation model to evaluate the effectiveness of the second part and loss function of D_1 and D_2 in VFLGAN. As shown by our experimental results, the performance of VFLGAN and VFLGAN-base are similar for low-dimensional data. However, for high-dimensional data like images in MNIST datasets, some information will be lost during the concatenation of intermediate features from different parties and the second part of D_1 and D_2 and their loss functions can help to improve synthetic data quality. VFLGAN-base is the typical structure of VFL, where one active party (D_s) calculates the loss function to update local models in passive parties. In our adjustment of VFLGAN, passive parties (party 1 and party 2) also contribute to their own local model updates.

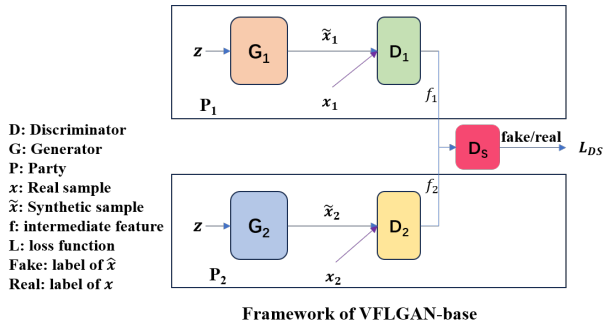


Figure 12: Framework of VFLGAN-base.

A.6 Supplemental Experiment Results: Normalization for Integer Categorical Attribute

In this section, we apply normalization (22) to preprocess the 'quality' attribute of red-wine-quality and white-wine-quality datasets. The FD curves during training are shown in Fig. 13. We can see that the FD curves of the proposed VFLGAN and VFLGAN-base get very close to the WGAN while the FD curve of VertiGAN is above that of WGAN with a significant gap. The AI training utility is shown in Table 5 and the proposed methods have significantly superior performance compared to VertiGAN. Besides, we can see that the AI utilities in Table 5 are lower than those of corresponding methods in Table 1, which means one-hot representation is more suitable for integer categorical attributes.

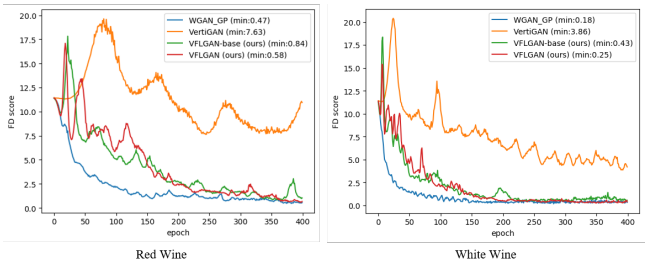


Figure 13: FD curves on the Red-wine-quality dataset (the lower the better).

A.7 Vulnerable Record Identification

In [47], the authors select records with the most outlier attributes as the most vulnerable records. There may be multiple records that have the most outlier attributes. Specifically, in our implementation, we select outliers with the following process. First, we select the value of the first quartile and the third quartile of an attribute denoted as Q_1 and Q_4, respectively. Then, we calculate the threshold by $T = Q_3 - Q_1$. Last, we determine the attribute a of a record as an outlier if it satisfies one of the following conditions,

$$\text{condition 1:} \quad Q_1 - a > T, \tag{30}$$

$$\text{condition 2:} \quad a - Q_3 > T. \tag{31}$$

In [38], the authors calculate the distance between a record and its nearest neighbour record and select the record with the maximum distance as the most vulnerable record. There is usually only one record that has the maximum nearest neighbour distance. Specifically, the distance is calculated by,

$$\begin{aligned} d\left(x_i, x_j\right) :=& 1-\frac{\left|\mathcal{F}_{\text{cat}}\right|}{F} \frac{h\left(x_i\right) \cdot h\left(x_j\right)}{\left\|h\left(x_i\right)\right\|_2 * \left\|h\left(x_j\right)\right\|_2} \\ & -\frac{\left|\mathcal{F}_{\text{cont}}\right|}{F} \frac{c\left(x_i\right) \cdot c\left(x_j\right)}{\left\|c\left(x_i\right)\right\|_2 * \left\|c\left(x_j\right)\right\|_2}, \end{aligned} \tag{32}$$

where F is the number of attributes, $|\mathcal{F}_{\text{cat}}|$ and $|\mathcal{F}_{\text{cont}}|$ denote the number of categorical attributes and the number of continuous attributes, respectively, $h(x)$ denotes the one-hot encoding of categorical attributes, and $c(x)$ denotes the vector of continuous attributes.

In Table 2, we select the most vulnerable record with the method of [38] as Record 1 and select the most vulnerable record with the method of [47] as Record 2. Interestingly, Record 1 of the Adult dataset has both the maximum nearest neighbour distance and the most outlier attributes.

A.8 Summary of Notation

All the symbols and notations used in the proposed scheme are defined in Table 6.

Table 6: Symbols and Notations

Notation	Description
DP	differential privacy
RDP	Rényi differential privacy
VFL	vertical federated learning
HFL	horizontal federated learning
GAN	generative adversarial network
ASSD	Auditing Scheme for Synthetic Datasets
ASIF	Auditing Scheme for Intermediate Features
P_v	distribution of variable v
x/X	real data / real dataset
x_i/X_i	real data / real dataset of party i
$\tilde{x}/\tilde{X}$	synthetic data / synthetic dataset
$\tilde{x}_i/\tilde{X}_i$	synthetic data / synthetic dataset of party i
λ	balancing coefficient
η	learning rate
B	size of the mini-batch during training
x^B	a mini-batch of real samples
$\tilde{x}^B$	a mini-batch of synthetic samples
D_i	discriminator i (discriminator of party i)
D_i^1	the first part of discriminator i
D_s	shared discriminator
G_i	generator i
$\mathcal{L}_M$	loss function of model M
θ_M	model M's parameters
θ_M^1	parameters of model M's first layer
$\theta_M^{2:n}$	parameters between M's 2^{nd} layer to n^{th} layer
$\mathcal{G}_M$	gradients of model M's parameters
$\mathcal{G}_M^1$	gradients of θ_M^1
$\mathcal{G}_M^{2:n}$	gradients of $\theta_M^{2:n}$