Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data

Xue Jiang
xue.jiang@tum.de
Technical University of Munich
Garching, Germany

Yufei Zhang
yufei.zhang@tum.de
Technical University of Munich
Garching, Germany

Xuebing Zhou
xuebing.zhou@huawei.com
Huawei Munich Research Center
Munich, Germany

Jens Grossklags
jens.grossklags@in.tum.de
Technical University of Munich
Garching, Germany

ABSTRACT

In the era of big data, user data are often vertically partitioned and stored at different local parties. Exploring the data from all the local parties would enable data analysts to gain a better understanding of the user population from different perspectives. However, the publication of vertically-partitioned data faces a dilemma: on the one hand, the original data cannot be directly shared by local parties due to privacy concerns; on the other hand, independently privatizing the local datasets before publishing may break the potential correlation between the cross-party attributes and lead to a significant utility loss. Prior solutions compute the privatized multivariate distributions of different attribute sets for constructing a synthetic integrated dataset. However, these algorithms are only applicable for low-dimensional structured data and may suffer from large utility loss with the increase in data dimensionality.

Following the idea of synthetic data generation, we propose VertiGAN, the first framework based on a generative adversarial network (GAN) for publishing vertically-partitioned data with privacy protection. The framework adopts a GAN model comprised of one multi-output global generator and multiple local discriminators. The generator is collaboratively trained by the server and local parties to learn the distribution of all parties' local data and is used to generate a high-utility synthetic integrated dataset on the server side. Additionally, we apply differential privacy (DP) during the training process to ensure strict privacy guarantees for the local data. We evaluate the framework's performance on a number of real-world datasets containing 68–1501 classification attributes and show that our framework is more capable of capturing joint distributions and cross-attribute correlations compared to statistics-based baseline algorithms. Moreover, with a privacy guarantee of $\epsilon = 8$, our framework achieves around a 2% $\sim$ 15% improvement in classification accuracy compared to the baseline algorithms. Extensive experimental results demonstrate the capability and efficiency of our framework in synthesizing vertically-partitioned data while striking a satisfactory utility-privacy balance.

Proceedings on Privacy Enhancing Technologies 2023(2), 236–250

https://doi.org/10.56553/popets-2023-0050

KEYWORDS

Differential privacy, vertically-partitioned data, synthetic data

1 INTRODUCTION

With the rapid development of network and computer technologies, large and diverse quantities of user data have been extensively collected and stored by different companies and institutes (referred to as local parties). These data usually contain rich information characterizing user profiles, which is valuable for data mining and building AI services. Due to the variety in service scenarios, the user data are often vertically partitioned and distributed among these local parties. That is, the local dataset held by each party usually contains different attributes of the same group of users. Considering that the more attributes the data consist of, the more information can be used for describing an individual user, it is practical for local parties to collaborate with each other and publish an integrated dataset with all the attributes for better decision making or building high-accuracy services. For instance, in a healthcare scenario, a group of specialist hospitals could publish a joint dataset to study potential correlations between different types of illnesses such as cancer, and heart and lung diseases. Similarly, in a smart finance scenario, a loan company could use a dataset jointly published by a bank and an e-commerce company to more deeply explore the key attributes that may result in higher default risk. More generally, integrating and analyzing these vertically-partitioned datasets enables data analysts to explore the hidden correlations of attributes from different perspectives and thus obtain a better understanding of the characteristics of user groups. This can be of significant help in designing optimized data mining algorithms and machine learning models.

However, publishing vertically-partitioned datasets has to be recognizant of the restrictions of data protection regulations such as the GDPR and users' privacy concerns. On the one hand, since the local data are generated based on users' ongoing behaviors and may contain sensitive information of individual users, directly sharing the original local datasets with an untrusted third party may lead to serious privacy leakage (see, for example, [5, 7]). On the other hand, the local parties can use state-of-the-art privacy-enhancing techniques, such as differential privacy (DP) [18], to process the real data and only share the privatized datasets. Nevertheless, each party individually privatizing the local data may break the correlations and joint distributions among attributes held by

different parties and lead to distinctive utility loss in the published dataset. Therefore, solutions for publishing vertically-partitioned data under a satisfactory privacy-utility balance are greatly needed.

In comparison to the substantial attention given to privacy-preserving data mining and machine learning under a vertical setting, algorithms for publishing the vertically-partitioned data are still barely studied. Prior works [26, 36] proposed two-party publication protocols under k-anonymity guarantees [52]. Unfortunately, later studies [51, 62] pointed out that k-anonymity models are vulnerable to various privacy attacks and cannot provide sufficient privacy protection. Follow-up work [44] proposed the first algorithm for publishing vertically-partitioned data under DP guarantees. However, the algorithm is limited to two-party scenarios and requires pre-defined taxonomy trees for all categorical attributes. Recent work by Tang *et al.* [53] proposed to use a latent tree model [70] to represent the cross-attribute distributions in the original dataset and privatizes the latent tree parameters via a distributed Laplace protocol to achieve ϵ-DP for each local dataset. Although the work by Tang *et al.* [53] effectively improves data utility and efficiency compared to [44], the algorithm evenly splits the privacy budget to all the attribute pairs. Therefore, the noise scale may increase exponentially with the data dimensionality and cause significant utility loss. Moreover, the algorithm is limited to discrete structured datasets and cannot support other data types.

In recent years, data synthesis has increasingly been considered a useful approach for addressing data insufficiency problems in developing AI applications. With the strong capabilities of characterizing the correlations and distributions of high-dimensional data, deep generative models such as generative adversarial networks (GANs) are increasingly used for generating high-utility and low-sensitivity synthetic data. Although some recent works (*e.g.*, [28, 54]) also proposed training the generative models under the federated learning (FL) framework to avoid the direct collection of real local data, the solutions all focus on the horizontal setting, which cannot be directly applied to vertically-partitioned data.

In this paper, we address this research gap and propose VertiGAN, the first GAN-based framework for privacy-preserving publication of vertically-partitioned data. The framework adopts a distributed GAN architecture, comprised of a global generator and multiple local discriminators. By using a collaborative training strategy, the global generator is trained without accessing the real local data. Moreover, we adopt a multi-output structure for the generator, which enables the model to directly learn the correlations and distributions of the attributes held by different local parties and generate synthetic integrated data. Finally, we inject DP perturbation during the training process, which ensures that the generator and the synthetic data satisfy strict DP guarantees for each local party. The main contributions of our approach are as follows:

- We propose VertiGAN, an efficient and privacy-preserving framework for publishing vertically-partitioned data. The framework trains a multi-output global generator to directly learn the distribution of all parties' local data and to generate high-utility synthetic integrated data on the server side. To the best of our knowledge, this is the first framework based on a deep generative model for private data publication under the vertical setting.
- We introduce a distributed training strategy, where the global generator is updated based on the gradients calculated by the local discriminators. The strategy eliminates the need to access real local data when training the global generator. Moreover, we apply DP perturbation during the training process to provide a strict privacy guarantee for each local dataset.
- We implement our framework and evaluate the performance on a number of real-world datasets containing 68–1501 classification attributes. Through comparison with the previous statistics-based algorithms, we show that the synthetic data generated by our framework always preserve much closer joint distributions and correlations to real data. Moreover, with a local privacy guarantee $\epsilon = 8$, we achieve around $2\% \sim 15\%$ improvement in classification accuracy compared to the baseline algorithms. Extensive evaluation experiments show that our framework has outperforming capability and efficiency in collecting high-dimensional data while offering a favorable utility-privacy balance.

2 RELATED WORK

2.1 Data Analysis on Vertically-Partitioned Data

In recent decades, data analysis on vertically-partitioned data has attracted increasing attention. Different from the horizontal setting, vertical partitioning refers to the scenario that local parties collect different attributes of the same set of users. Existing applications on vertically-partitioned data include, for instance, jointly training ML models using attributes of all the local parties, or publishing an integrated dataset for future data mining.

2.1.1 Machine Learning Under Vertical Setting. In the context of ML, prior studies by Vaidya *et al.* proposed a series of secure multi-party computation (SMC) protocols [66] for training different models on vertically-partitioned data, including Bayes classifier [55], and decision trees [56], etc. Hardy *et al.* [22] proposed a vertical federated learning (VFL) framework, which trained LR models using homomorphic encryption (HE) [14]. Yang [65] further applied the quasi-Newton method in VFL to reduce the number of communication rounds. Some other works [12, 63] also proposed solutions for tree-based models and neural networks [48]. Besides using crypto-based technologies such as HE and SMC to ensure security in VFL, recent works [11, 59] further proposed to incorporate DP into the training process to provide strict privacy guarantees for local data.

On the other hand, some recent works also investigate potential privacy attacks against VFL, which include label inference attacks and feature reconstruction attacks. In the label inference attacks, the parties without ground-truth labels aim to use the back-propagated gradients to infer the sample labels. Several existing attacks proposed to explore the difference of the gradient norms [39] or the sign of the last-layer gradients [40, 73]. Other research [19] also proposed a semi-supervised learning approach that first estimated the bottom-layer parameters and then used the "completed" model to "generate" the label of arbitrary samples. Apart from the label leakage, some other works [27, 41] also studied the feature leakage

in VFL, where the party obtaining the model predictions tries to reconstruct the input features of other parties. Nevertheless, existing attacks against VFL only focused on classification models, where the attackers either try to infer the ground-truth labels or need to use the model predictions to reconstruct local features. In contrast, in this paper, we use the GAN model for data synthesis, which does not involve such label (or prediction) information. Hence, the above-mentioned attacks in VFL are no more applicable.

2.1.2 Data Publication Under Vertical Setting. Compared to the extensive set of studies on machine learning under the vertical setting, there are still only limited prior works on publishing vertically-partitioned data. Prior works in [26, 36] proposed SMC-based protocols for two-party data publication under k-anonymity guarantees [52]. Nevertheless, later studies [51, 62] pointed out that k-anonymity models are vulnerable to various privacy attacks and cannot provide sufficient privacy protection. In contrast, DP [18] is considered as a more principled approach for private data publication. Mohammed *et al.* proposed DistDiffGen [44], the first algorithm for publishing vertically-partitioned data under DP guarantees. DistDiffGen first generalizes the raw data using a distributed exponential mechanism and then adds noise to the distributions to ensure ϵ-DP. However, the algorithm is limited to two-party scenarios and requires pre-defined taxonomy trees for all categorical attributes, which may not always be available in practice. Later work by Tang *et al.* [53] proposed an improved differentially private latent tree (DPLT) algorithm, which first uses a latent tree model [70] to represent the cross-attribute distributions in the original dataset and then privatizes the latent tree parameters via a distributed Laplace protocol to achieve ϵ-DP for each local dataset. The latent tree model will then be used for generating a synthetic dataset. Although [53] significantly improves the data utility and efficiency in comparison to [44], it is still limited to discrete attributes. Moreover, since the privacy budget is evenly split over all the attribute pairs, the noise scale may increase exponentially with the increased data dimensionality and cause a large utility loss.

In this paper, we propose a distributed GAN-based protocol for publishing vertically partitioned data in a private manner. Compared to previous works, our solution can support the publication of high-dimensional datasets with strict DP guarantees. Moreover, the framework can be further extended to support other types of data such as numerical data and images.

2.2 Differentially-Private Data Synthesis

DP data synthesis has been extensively studied over recent years as one of the solutions for privacy-preserving data publishing. Previous statistics-based works [38, 68] computed joint distributions of original structured data under DP guarantees and used them to generate synthetic datasets. However, these methods can only be applied to structured data and may suffer from a significant utility loss with the increase in data dimensionality.

Inspired by the rapid evolution of deep learning, later works proposed to directly train generative models such as autoencoders [3, 35] and generative adversarial networks (GANs, [20]) and to generate high-utility synthetic data. Nevertheless, simply training these generative models without protection may still lead to privacy leakage. For instance, prior work [4, 57] showed that GANs may unintentionally memorize the training data. Moreover, Hayes *et al.* [23] proposed different membership inference attacks against the trained generator and discriminators. Later works also demonstrated that the membership information can be revealed from the generated synthetic data [10, 24, 50]. In addition, Zhou *et al.* [72] performed a property inference attack, which uses the released synthetic data to infer the macro-level information of training data (*e.g.*, the ratio of samples regarding a certain property).

DP has been considered one of the countermeasures against such privacy attacks. Existing DP data synthesis algorithms are generally divided into two categories, namely by using differentially-private stochastic gradient descent (DPSGD, [1]) or private aggregation of teacher ensembles (PATE, [45]). The DPSGD-based algorithms [64, 71] perturb the model gradients in each iteration by clipping and adding Gaussian noise to ensure DP guarantees. The PATE-based algorithms [31, 58] first train a group of teacher models (*e.g.*, the discriminator in GAN) on non-overlapping subsets of original data and then use the noisy predictions from the teacher group to train the student model (*e.g.*, the generator). Nevertheless, previous data synthesis algorithms mainly focus on the centralized setting, where the server has already collected the clients' real data. This may not always be realistic since the clients may refuse to share their personal local data with untrusted servers. Therefore, some recent works also proposed to train the generative autoencoders [28] and GANs [54, 69] under the FL framework to avoid the collection of original data. However, existing solutions only focus on the horizontal setting, where the local data shares the same set of attributes. In contrast, in this paper, we conduct the first attempt at the GAN-based DP data synthesis for vertically-partitioned data.

3 BACKGROUND

3.1 Differential Privacy

DP [18] is a state-of-the-art anonymization technique that provides rigorous privacy guarantees for data analysis. The classic definition of DP is as follows:

Definition 1 ((ϵ, δ)-DP [18]). *A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-DP if for any two adjacent datasets $\mathcal{X}, \mathcal{X}'$ differing in one data sample and any measurable subset of outputs $\mathcal{Y} \subseteq range(\mathcal{M})$ we have*

$$\Pr[\mathcal{M}(\mathcal{X}) \in \mathcal{Y}] \leq e^{\epsilon} \cdot \Pr\left[\mathcal{M}(\mathcal{X}') \in \mathcal{Y}\right] + \delta, \tag{1}$$

where ϵ is the privacy loss and δ is the probability of privacy leakage. When $\delta = 0$, we have ϵ-DP.

The original DP defined an upper bound of the privacy cost. Recent works further proposed various relaxations of DP to achieve tighter bounds for the privacy cost, especially for iterative algorithms. One of the widely used definitions is Rényi DP (RDP) [43], which uses the Rényi divergence to measure the distance between two probabilities. The definition of RDP is as follows:

Definition 2 ($(\alpha, \epsilon(\alpha))$-RDP [43]). *A randomized mechanism $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP if for any two adjacent datasets $\mathcal{X}, \mathcal{X}'$ differing in one data sample, the Rényi α-divergence between $\mathcal{M}(\mathcal{X})$ and $\mathcal{M}(\mathcal{X}')$ satisfies*

$$\mathcal{D}_{\alpha}(\mathcal{M}(\mathcal{X}) || \mathcal{M}(\mathcal{X}'))) \triangleq \frac{1}{\alpha - 1} \log \mathbb{E}\left[\left(\frac{\mathcal{M}(\mathcal{X})}{\mathcal{M}(\mathcal{X}')}\right)^{\alpha}\right] \leq \epsilon. \tag{2}$$

Similar to DP, a Gaussian mechanism can also be used to achieve $(\alpha, \epsilon(\alpha))$-RDP:

Definition 3 (Gaussian mechanism). *For a real-valued function $f : \mathcal{X} \to \mathbb{R}^d$ with l_2 sensitivity Δ_f defined as*

$$\Delta_f = \max_{\mathcal{X},\mathcal{X}'} ||f(\mathcal{X}) - f(\mathcal{X}')||_2 \tag{3}$$

over all adjacent datasets $\mathcal{X}$ and $\mathcal{X}'$. The following Gaussian mechanism $\mathcal{M}_\sigma$ satisfies $(\alpha, \epsilon(\alpha))$-RDP:

$$\mathcal{M}_\sigma(x) = f(x) + \mathcal{N}(0, \sigma^2 I), \text{ where } \epsilon(\alpha) = \frac{\Delta_f^2 \alpha}{2\sigma^2}. \tag{4}$$

Moreover, RDP also preserves the composition property for accumulating the privacy cost over a sequence of mechanisms. Namely:

Theorem 1 (Composition property). *Suppose n mechanisms $\{\mathcal{M}_1, \cdots, \mathcal{M}_n\}$ respectively satisfy $(\alpha, \epsilon_i(\alpha))$-RDP, and are sequentially computed on the same set of private data $\mathcal{X}$, then a mechanism formed by $(\mathcal{M}_1, \cdots, \mathcal{M}_n)$ satisfies $(\alpha, \sum_{i=1}^n \epsilon_i(\alpha))$-RDP.*

Theorem 2 (Robustness to post-processing). *Let $\mathcal{M}$ be an $(\alpha, \epsilon(\alpha))$-RDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ also satisfies $(\alpha, \epsilon(\alpha))$-RDP.*

Additionally, the accumulated privacy cost under RDP can be further amplified by the subsampled mechanism:

Lemma 1 (RDP for Subsampled Mechanism [60]). *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 $m \le n$, let the randomized algorithm $\mathcal{M} \circ$ `subsample` be defined as: (1)* `subsample`*: subsample without replacement m data points of the dataset (sampling parameter $\gamma = m/n$), and (2) apply $\mathcal{M}$: a randomized algorithm taking the subsampled dataset as the input. For all integers $\alpha \ge 2$, if $\mathcal{M}$ obeys $(\alpha, \epsilon(\alpha))$-RDP, then the new randomized algorithm $\mathcal{M} \circ$ `subsample` obeys $(\alpha, \epsilon'(\alpha))$-RDP where*

$$\begin{aligned}\epsilon'(\alpha) \le \frac{1}{\alpha - 1} \log \Bigg(1 + \gamma^2 \binom{\alpha}{2} \min \big\{4(e^{\epsilon(2)} - 1), \\ e^{\epsilon(2)} \min\{2, (e^{\epsilon(\infty)} - 1)^2\}\big\} \\ + \sum_{j=3}^{\alpha} \gamma^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min\{2, (e^{\epsilon(\infty)} - 1)^j\}\Bigg).\end{aligned} \tag{5}$$

Finally, the privacy guarantees under RDP can be converted to the original DP guarantees:

Lemma 2 (RDP to DP [43]). *If a mechanism $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP, then $\mathcal{M}$ satisfies $(\epsilon(\alpha) + \frac{\log 1/\delta}{\alpha - 1}, \delta)$-DP for any $\delta \in (0, 1)$.*

3.2 Generative Adversarial Network

The GAN [20] is a class of unsupervised learning algorithms that have been extensively studied in the last decade due to its strong capability in generating high-fidelity synthetic data. A GAN model usually consists of a generator G and a discriminator D. The generator G takes as input a random noise z from a certain latent distribution P_z and generates synthetic data $\tilde{x} = G(z)$. The discriminator D learns to distinguish between data drawn from the real distribution $x \sim P_r$ and from the synthetic distribution $\tilde{x} \sim P_g$, where P_g is determined by G and P_z. This can be considered a binary classification task. Both models are trained simultaneously through an adversarial process, where the generator keeps improving the quality of the synthetic data to fool the discriminator while the discriminator tries to discriminate between real and synthetic data with high accuracy. The ultimate goal is to approximate the real distribution P_r with the synthetic distribution P_g such that the discriminator cannot correctly distinguish between the real and the synthetic data. The problem can be formulated as a min-max training process with the following objective [20]:

$$\mathcal{L}_{GAN} = \mathop{\mathbb{E}}_{x \sim p_r}[\log D(x)] + \mathop{\mathbb{E}}_{\tilde{x} \sim p_g}[\log(1 - D(\tilde{x}))], \tag{6}$$

where P_r is the distribution of real data and P_g is the distribution of synthetic data $\tilde{x} = G(z)$ with $z \sim P_z$.

By utilizing different generator and discriminator structures, GANs have been adjusted to generate various types of synthetic data such as tabular data [46], images [33], and time-series data [67]. Nevertheless, the original GAN models usually suffer problems such as training instability and failure to converge. Therefore, some other works proposed to modify the loss function to improve the model convergence. The Wasserstein GAN (WGAN) [3, 61] is one of the well-known improved GANs. In comparison with the original loss function, WGAN-GP uses the Wasserstein-1 distance with an additional gradient norm penalty to achieve Lipschitz continuity. Given the real data x, the input noise $z \sim P_z$ and the synthetic data $\tilde{x} = G(z)$, the gradient penalty term can be written as

$$(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2, \quad \text{where} \quad \hat{x} = \mu x + (1 - \mu)\tilde{x}. \tag{7}$$

Here $\hat{x}$ is a weighted average between the real and synthetic data and $\mu \sim \mathbf{U}(0, 1)$ is a randomly sampled weight. Thus, the loss function for the generator and discriminator is formulated as follows:

$$\mathcal{L}_G = D(\tilde{x}) = D(G(z)), \tag{8}$$

$$\mathcal{L}_D = D(x) - D(\tilde{x}) + \lambda(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2, \tag{9}$$

where λ is the weight for the gradient penalty.

In this paper, we choose P_z to follow the standard Gaussian distribution $\mathcal{N}(0, I)$ and $\lambda = 10$ for the gradient penalty. Similar to Equation (6), the loss function of WGAN can be formulated as:

$$\mathcal{L}_{WGAN} = \mathop{\mathbb{E}}_{\tilde{x} \sim p_g}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x \sim p_r}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x} \sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]. \tag{10}$$

4 PROBLEM STATEMENT

In this paper, we focus on the scenario where the user data are vertically partitioned and distributed over multiple local parties. Each party possesses a different set of attributes of the same group of samples. A central server aims to integrate these local datasets in a private manner and publish a joint dataset containing all the attributes. The joint dataset will be further used by external data analysts for downstream data mining and model training tasks.

An illustration of the system setting is shown in Figure 1. We assume there are M local parties $\mathcal{P}_1, \cdots, \mathcal{P}_M$. Each party $\mathcal{P}_i$ has a local dataset containing a *different* set of attributes $A^i = \{a_1^i, \cdots, a_{|A^i|}^i\}$. Here, the attribute sets can be either partially overlapping or non-overlapping. Moreover, each party may hold samples not

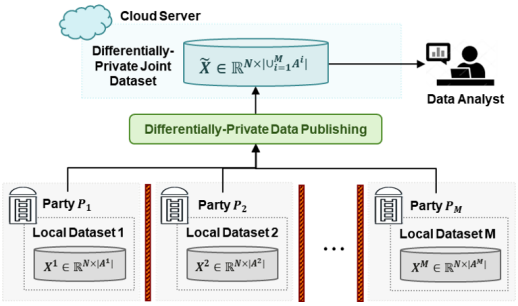

Figure 1: Overview of the system model.

covered by other parties. Therefore, we assume that the local data have certain alignable sample IDs (e.g., ID number, cellphone number, etc.). The local parties can use private set intersection (PSI) protocols (*e.g.*, [13, 15, 25]) to determine the intersecting sample IDs without exposing the non-intersect samples. Then, each party sorts the common samples according to their IDs and obtains the final training dataset $X^i \in \mathbb{R}^{N\times|A^i|}$, where N is the number of samples and $|A^i|$ is the number of attributes.

The goal of the task is to design a privacy-preserving framework, where a central server can collaborate with all the local parties and publish a private joint dataset $\tilde{X} \in \mathbb{R}^{N\times|\cup_{i=1}^{M} A^i|}$ that contains the full set of attributes. The joint dataset $\tilde{X}$ preserves both single-party and cross-party attribute correlations. More specifically, consider local parties $\mathcal{P}_i$ and $\mathcal{P}_j$ respectively holding local datasets $X^i \in \mathbb{R}^{N\times|A^i|}$ and $X^j \in \mathbb{R}^{N\times|A^j|}$, then the distribution of $\tilde{X}$ should satisfy

$$P_{\tilde{X}}(A^i) \approx P_{X^i}(A^i), \quad P_{\tilde{X}}(A^i, A^j) \approx P_{X^i,X^j}(A^i, A^j). \tag{11}$$

Following previous works, we assume that the local parties and the central server are *honest-but curious*, who correctly follow the protocols but try to infer sensitive information of other local datasets. Moreover, we also consider the threat posed by external data analysts, who aim to use the published joint dataset to re-identify sensitive information of specific users. Based on the considerations above, it is required that there is no information exchange among local parties and each party does not know the attribute set of other parties. Moreover, we assume that the server cannot directly access the raw local data but is aware of the full attribute set and the size of the training dataset. Finally, the published dataset should satisfy strict DP guarantees and not reveal the privacy of individual users in the local datasets.

5 PROPOSED FRAMEWORK

Although previous works proposed statistics-based algorithms for publishing vertically-partitioned data under DP guarantees, the solutions are only limited to low-dimensional structured data and may suffer from large utility loss with the increase in domain size. Following the idea of data synthesis, we propose VertiGAN, the first GAN-based framework for differentially-private publication of vertically-partitioned data. The overall workflow of the framework is presented in Figure 2, which consists of two phases, namely the *collaborative training process* and the *synthetic data generation process*. In the first process, a GAN model is collaboratively trained by the server and all the local parties to learn the correlations and distributions of all the local datasets in a private manner. In the second phase, the generator part is used to directly generate synthetic integrated data that contains attributes held by all the local parties. The synthetic data preserves similar statistical properties to real data and can be alternatively used for downstream data analysis and AI training tasks.

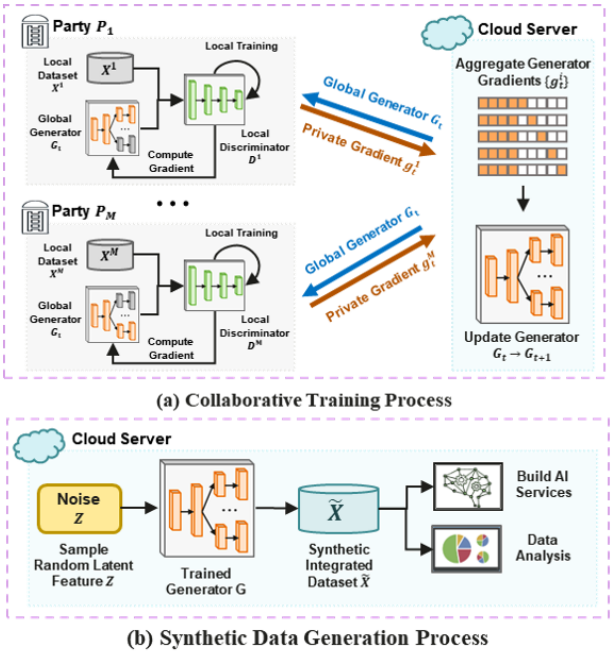

Figure 2: General workflow of the VertiGAN framework.

Nevertheless, training the GAN model on distributed vertically-partitioned data faces several challenges. To start with, in this paper, we focus on the scenario where the real data are distributed on the local side and cannot be directly shared with the server. Hence, the model cannot be simply trained as in the centralized setting due to *data inaccessibility*. Moreover, in the vertical setting, the attribute sets held by the local parties are usually different from each other, which is referred to as *attribute inconsistency* in this paper. This causes existing solutions that train GANs in the horizontal FL framework to be inapplicable. Finally, recent contributions (*e.g.*, [9, 49]) point out that the ML models may memorize information in training data and suffer from different privacy attacks. Therefore, *privacy protection* techniques should be applied during model training to prevent potential privacy leakage. We apply corresponding solutions in the VertiGAN framework to address the above-mentioned challenges. In the following sections, we will respectively introduce each solution in detail.

5.1 Distributed GAN Against Data Inaccessibility

Different from other generative models, GANs are usually built with two independent networks, namely a generator and a discriminator.

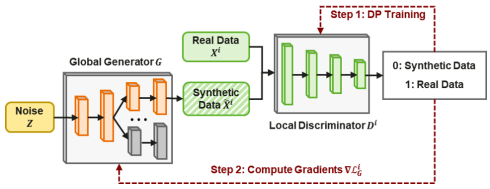

Figure 3: Workflow of the local training process.

The two networks are trained in an adversarial manner to improve their own performance. By taking advantage of GANs' separate generator-discriminator architecture, the VertiGAN framework applies a distributed training strategy to address *data inaccessibility* problems. More specifically, the framework deploys a global generator G on the server side and multiple discriminators $\{D^1, \cdots, D^M\}$ on the local side. The global generator takes in random latent features and outputs synthetic data for each local party, while the local discriminators are trained on the local side to distinguish between real data and synthetic data. The ultimate goal of the framework is to obtain a well-trained global generator on the server side that is capable of producing high-utility synthetic data without violating the privacy of real local data.

The training process is conducted in cooperation with the server and all the local parties, as shown in Figure 2. During each global training round, the server broadcasts the current global generator to all the local parties for generating synthetic data. Each party first uses its real local data and the corresponding part of synthetic data to train its local discriminator and then uses the trained discriminator to compute the generator's gradient. Finally, the gradients from all the local parties will be aggregated on the server side and used to update the global generator. In Figure 3, we also present a detailed illustration of the local training process. It can be seen that the local data are only used for training the local discriminator D^i, and the global generator G is only updated based on the gradient computed by the trained discriminators. Moreover, only the information (weights and gradients) of the generator is exchanged between the local and server side, while the discriminators and the real data are always kept on the local side. In this way, the framework can facilitate the training of the global generator without direct access to the real local data.

5.2 Multi-Output Generator Against Attribute Inconsistency

Moreover, in this paper, we consider the scenario where the user data are vertically-partitioned and distributed among M local parties. Since the local parties under this setting may hold different sets of attributes, the conventional single-output generators are not applicable for the framework. In order to address the *attribute inconsistency* problem, we propose a multi-output structure for the global generator. The generator consists of several common layers (denoted as G^0) and M separate follow-up branches (denoted as $\{G^1, \cdots, G^M\}$). Each branch G^i produces synthetic data with attributes of one local party $\mathcal{P}_i$. Given a batch of input feature Z, the global generator is capable of concurrently producing synthetic data $\{\tilde{X}^1, \cdots, \tilde{X}^M\}$ for all the local parties. Here, $\tilde{X}^i = G^i(G^0(Z))$ corresponds to the data generated from the i-th branch.

We follow the optimization approach of WGAN introduced in Section 3.2 to iteratively train the global generator and local discriminators in the proposed framework. On the one hand, the training of the discriminators on the local side is conducted as under the centralized setting. Here, the loss function for the i-th local discriminator D^i is:

$$\mathcal{L}_D^i = D^i(x^i) - D^i(\tilde{x}^i) + \lambda(\|\nabla_{\hat{x}^i} D(\hat{x}^i)\| - 1)^2, \tag{12}$$

where x^i is the real data of the i-th local party, $\tilde{x}^i = G^i(G^0(z))$ is the synthetic data generated by the i-th branch of G, $\hat{x}^i$ is the gradient penalty as defined in Equation (7), and λ is the weight for the gradient penalty. Once the discriminators have been trained for several iterations, they will be used to compute the gradient of the global generator. The loss function for the global generator G can be computed as the sum of the loss regarding all the local discriminators, each of which is derived following Equation (8):

$$\mathcal{L}_G = \sum_{i=1}^{M} \mathcal{L}_G^i = \sum_{i=1}^{M} D^i(G^i(G^0(z))). \tag{13}$$

The generator's gradient $\nabla\mathcal{L}_G$ can be further derived as

$$\begin{aligned}
\nabla\mathcal{L}_G &= \frac{\partial \sum_{i=1}^{M} \mathcal{L}_G^i}{\partial G} = \sum_{i=1}^{M} \frac{\partial \mathcal{L}_G^i}{\partial [G^0, G^1, \cdots, G^M]} \\
&= \left[\frac{\partial \mathcal{L}_G^1}{\partial G^0}, \frac{\partial \mathcal{L}_G^1}{\partial G^1}, 0, \cdots, 0\right] + \cdots + \left[\frac{\partial \mathcal{L}_G^M}{\partial G^0}, 0, 0, \cdots, \frac{\partial \mathcal{L}_G^M}{\partial G^M}\right] \\
&= \left[\sum_{i=1}^{M} \frac{\partial \mathcal{L}_G^i}{\partial G^0}, \frac{\partial \mathcal{L}_G^1}{\partial G^1}, \cdots, \frac{\partial \mathcal{L}_G^M}{\partial G^M}\right]
\end{aligned} \tag{14}$$

which is the sum of the generator gradients from all the local parties. Hence, by aggregating all the returned generator gradients, the server achieves to use the sum of the gradients to update the global generator. It can be seen from Equation (14) that the parameters of each branch G^i are updated based on the gradients from party $\mathcal{P}_i$, while the parameters of the common layers G^0 are updated by the gradients from all the local parties. Therefore, the multi-output structure enables the global generator to automatically capture the correlations and distributions of attributes across local parties during the training process and directly generate synthetic integrated data with the entire attribute set.

5.3 Collaborative Training with DP

In the previous sections, we illustrate how the VertiGAN framework enables a global generator to learn the hidden correlations of attributes across all the local parties without actually accessing the real local data. Nevertheless, recent studies (*e.g.*, [9, 49]) showed that the trained generator may reveal sensitive information of real local data under various privacy attacks. In order to mitigate potential privacy risks, we further apply DP during the training process, which provides strict privacy guarantees to the local datasets.

Considering the global generator does not directly access real local data, we follow previous DP-GAN algorithms [64, 71] and only perturb the gradients of local discriminators to achieve privacy protection. Specifically, in each update step of the discriminator, we first sample a batch of real local data and synthetic data, and then

compute the corresponding gradients $\{g_D^{i,b}\}_{b\in B}$. Each gradient $g_D^{i,b}$ is then clipped by a pre-defined L_2-norm bound C, namely

$$\bar{g}_D^{i,b} = \texttt{clip}(g_D^{i,b}, C) = g_D^{i,b}/\max(1, ||g_D^{i,b}||_2/C). \tag{15}$$

Next, we sum up all the clipped gradients, add random Gaussian noise $\mathcal{N}(0, \sigma^2C^2I)$, and divide the perturbed gradient by the batch size B as shown below:

$$\tilde{g}_D^i = \frac{1}{B}\left(\sum_{b=1}^{B} \bar{g}_D^{i,b} + \mathcal{N}(0, \sigma^2C^2I)\right). \tag{16}$$

The gradient $\tilde{g}_D^i$ is used to update the discriminator parameters.

Since the local discriminator is repeatedly updated during the training process, according to the composition property, the total privacy cost should be accumulated. Considering that RDP achieves a much tighter privacy estimation in comparison to the traditional DP (as mentioned in Section 3.1), we first compute the overall privacy cost under the RDP definition and then convert it back to the traditional DP definition. To start with, the privacy cost of each gradient perturbation under RDP is derived as follows:

Corollary 1. *With a noise scale $\mathcal{N}(0, \sigma^2C^2)$, the perturbed gradient $\tilde{g}_D^i$ satisfies $(\alpha, \alpha/2\sigma^2)$-RDP.*

Proof. Let $f = \sum_{b=1}^{B} \bar{g}_D^b = \sum_{b=1}^{B} \texttt{clip}(g_D^b, C)$ be the sum of all the gradients clipped by an L_2-norm bound of C. The sensitivity of f can be derived as:

$$\Delta_f = \max_{\mathcal{D},\mathcal{D}'} ||f(\mathcal{D}) - f(\mathcal{D}')||_2 \leq C. \tag{17}$$

Furthermore, the gradient perturbation process can be denoted as $\mathcal{M}_f = f + \mathcal{N}(0, \sigma^2C^2I)$. Based on Section 3.1, the privacy cost of $\mathcal{M}_f$ under the order α is

$$\epsilon(\alpha) = \frac{(\Delta_f)^2 \cdot \alpha}{2 \cdot \sigma^2C^2} = \frac{C^2 \cdot \alpha}{2 \cdot \sigma^2C^2} = \frac{\alpha}{2\sigma^2}. \tag{18}$$

As shown in Equation (16), the perturbed gradient will be divided by a batch size B, and the result $\tilde{g}_D^i$ will be actually used to update the discriminator. Since B is unrelated to the real data, according to the post-processing property (Theorem 2), the final discriminator update $\tilde{g}_D^i$ also satisfies $(\alpha, \epsilon(\alpha))$-RDP. □

According to Lemma 1, the privacy guarantee can be further amplified by subsampling. Given N as the total number of training data and B as the batch size, we compute the sampling rate as $\gamma = N/B$ and derive the amplified privacy cost $\epsilon'(\alpha)$ following Equation (5). Next, assume the discriminator has updated for T steps during the entire training process, then the overall privacy cost is $(\alpha, T \cdot \epsilon'(\alpha))$-RDP. We further convert privacy cost back to the traditional (ϵ, δ)-DP definition according to Lemma 2. Finally, since the global generator is trained on the local discriminators, according to the post-processing property (Theorem 2), the global generator also satisfies (ϵ, δ)-DP for the corresponding local dataset.

5.4 Overall Training Process

With the above design considerations, we now describe the overall training process presented in Algorithm 1 and Algorithm 2.

Before the training starts, the server initializes the global generator G. On the local side, each party also initializes its local discriminator D^i. Moreover, considering that the local parties may have personalized privacy requirements, we let each local party individually compute the noise scale σ^i. With the universally configured batch size B, global rounds T_{global}, and local steps T_d, the discriminator's total update step is derived as $T = T_{global} \cdot T_d$. Following the privacy accounting process described in Section 5.3, the required σ^i under the target privacy budget (ϵ^i, δ^i) can be determined accordingly. Finally, since each local party holds different attributes of the same group of samples, the local training data should be sample-wise aligned during each global round. A naive solution is to let the server randomly sample multiple batches of data indices for selecting the real data as well as input features for generating the synthetic data, and then broadcast all the information to the local side. However, this may cause extra communication costs, especially for large training batches. To address the issue, our framework applies a pseudorandom number generator (PRNG) Φ^i at *each* local party to realize the data alignment. Following prior works [6, 42], we use secure PRNGs to achieve comprehensive security guarantees. Moreover, we require that all the local PRNGs use the same algorithm and are deployed with the same configuration. Therefore, according to the reproducibility of PRNG, given the same random seed, each Φ^i is able to produce the same sequence of indices of real data or input features sampled from the standard Gaussian distribution. By using the PRNG, the server only needs to randomly sample a random seed and broadcast it to all the local parties in each global round, which significantly improves communication efficiency. Also, considering that existing secure PRNGs based on standard cryptographic primitives can have an output rate of gigabytes per second on modern CPUs [32], their computation cost is negligible compared to the local training time.

In each global training round, the server broadcasts the current global generator G as well as the random seed τ to all the local parties. Each party $\mathcal{P}^i$ first sets Φ^i with the random seed τ and then updates the local discriminator D^i for T_d steps using the real data X^i and the synthetic data $\tilde{X}^i = G^i(G^0(Z))$ sampled by Φ^i. We

Algorithm 1: VertiGAN - Workflow of Server

Input: G: global generator; M: number of local parties; T_{global}: global training rounds; η: learning rate; OPT: optimizer for the GAN model.
Output: Trained global generator G

Server executes:

Initialize global generator G
for each local party $i = 1, \cdots, M$ **do**
  LocalInitialization() `// Run on the local side`
end for
for each global round $t = 1, \cdots, T_{global}$ **do**
  Sample random seed τ
  for local party $i = 1, \cdots, M$ **do**
    Distribute τ and G to the local party i
    Get local gradient g_G^i = **LocalUpdate**(τ, G)
  end for
  Aggregate local gradients $g_G = \sum_{i=1}^{M} g_G^i$
  Update generator $G \leftarrow$ `OPT.update`(G, g_G, η)
end for
return G

Algorithm 2: VertiGAN - Workflow of Local Party i

Input: G: global generator; D^i: party i's local discriminator; X^i: party i's local data; Φ^i: party i's local PRNG; T_{global}: global training rounds; T_d: discriminator's local update steps; B: batch size; η: learning rate; C: L_2 clipping bound; (ϵ^i, δ^i): party i's privacy budget; OPT: optimizer for the GAN model.

LocalInitialization():

Initialize local discriminator D^i, local PRNG Φ^i
Given the target (ϵ^i, δ^i) and the pre-defined (B, T, T_d), compute the required noise scale σ^i

LocalUpdate(τ, G):

Get local data X^i, set Φ^i.set_seed(τ)
// Train local discriminator
for $t = 1, \cdots, T_d$ **do**
  Sample indices $J = \Phi^i$.random_choice(size=B)
  Sample input noise $Z = \Phi^i$.random_normal(size=B)
  for $b = 1, \cdots, B$ **do**
    Let $x = X^i[J[b]]$, $\tilde{x} = G^i(G^0(Z[b]))$
    Compute $\mathcal{L}_D^{i,b}(x, \tilde{x})$ and $g_D^{i,b} = \nabla \mathcal{L}_D^{i,b}(x, \tilde{x})$
    Clip gradient $\bar{g}_D^{i,b} = g_D^{i,b} / \max(1, ||g_D^{i,b}||_2 / C)$
  end for
  Aggregate gradients and add noise $\tilde{g}_D^i = \frac{1}{B}(\sum_{b=1}^B \bar{g}_D^{i,b} + \mathcal{N}(0, \sigma^{i\,2} C^2 I))$
  Update discriminator $D^i \leftarrow$ OPT.update($D^i, \tilde{g}_D^i, \eta$)
end for
// Compute generator gradient
Sample input noise $Z = \Phi^i$.random_normal(size=B)
Compute $g_G^i = \frac{1}{B} \sum_{b=1}^B \nabla \mathcal{L}_G(G^i(G^0(Z[b])))$
return g_G^i

apply the DP perturbation in each update step, where the batch of gradients is clipped by L_2 bound C and perturbed with random Gaussian noise $\mathcal{N}(0, \sigma^{i\,2} C^2 I)$. The noise scale σ^i is determined in the initialization process. Then, the local discriminator is used to compute the gradient g_G^i of the current global generator, which will be returned to the server for updating the parameters of the global generator parameters. The global training process is conducted for T_{global} rounds. Once the training completes, the server can use the global generator G to directly generate the synthetic dataset with attributes of all the local parties.

6 EXPERIMENTS AND RESULTS

We implemented the proposed framework using the Tensorflow library and performed comprehensive experiments with a number of open-source datasets to evaluate its performance. In this section, we first introduce the experimental settings and then discuss the evaluation results.

6.1 Experiment Setup

6.1.1 Datasets and Models. We used six multi-dimensional classification datasets for evaluating the performance of the VertiGAN framework:

Table 1: Datasets details

Dataset	Type	Num. Records	Num. Attributes	Domain Size
Census	Integer	2,458,285	68	2^{150}
Twitter	Integer	140,707	78	2^{181}
Web	Binary	36,974	124	2^{124}
Vehicle	Binary	98,528	101	2^{101}
HAR	Binary	10,299	561	2^{561}
Dilbert	Binary	10,000	1501	2^{1501}

Table 2: One-hot dimensions and the number of model parameters under the two-party setting

	Party 1		Party 2		Server
Dataset	**One-hot Dim.**	**#Param. D^1**	**One-hot Dim.**	**#Param. D^2**	**#Param. G**
Census	137	9,592	145	10,732	53,760
Twitter	195	19,307	174	15,313	94,768
Web	124	7,813	123	7,751	39,416
Vehicle	100	5,101	102	5,305	26,857
HAR	562	158,485	566	160,745	812,949
Dilbert	1,500	788,551	1,505	794,190	2,681,446

Web [47] contains records with 124 binary attributes extracted from each web page. The goal was to train a classifier to determine whether the web page belongs to a category.

Vehicle [17] contains data collected in wireless distributed sensor networks. Each record has 100 binary attributes representing data collected from different acoustic and seismic sensors. The goal was to train a classifier for vehicle type classification.

Census [16] contains records drawn from the 1990 United States census data, including 68 personal attributes such as gender, income, and marital status. We used the dataset to classify the duration of people's active duty service.

Twitter [34] contains records with 77 attributes such as the number of discussions and average discussion length, which are used to predict the popularity magnitude of each instance. In our experiment, we quantified the values of each attribute into five bins. The goal was to classify the level of popularity of each instance.

Activity [2] contains sensor records describing six daily activities. Each data record has 561 attributes representing different time and frequency domain variables. We normalize each attribute and convert the data to binary form.

Dilbert was originally provided in [37] for object recognition. We use the processed version in [21], where the records are categorized to five classes. We take the first 1500 attributes from the processed data to exclude the irrelevant variables mentioned in [21]. Then, we normalize each attribute and convert the data to binary form.

Details of each dataset are presented in Table 1, including the data type, the number of records and attributes, and the domain size. In the experiments, we assume that each party holds 10^5 data records. To this end, we randomly sample 10^5 records from each original dataset and partition the datasets by feature. If the original dataset

contains fewer records, the data are sampled with replacements. We further use one-hot encoding to convert the original categorical attributes to the numerical form for model training.

We design the global generator and local discriminators as multi-layer neural networks (NNs) and determine their layer size according to the one-hot dimension of the local datasets. The local discriminators are two-layer NNs whose output is a scalar between 0 and 1. The global generator is a multi-output model, which has two common layers followed by a number of separate branches. Each branch contains two fully-connected layers, which outputs the synthetic data of one party. In Table 2, we report the one-hot dimensions and the model size under the two-party setting.

6.1.2 Baseline Methods. Considering the objective and setting of existing works on the publication of vertically-partitioned data, we use the DPLT algorithm proposed by Tang *et al.* [53] as our baseline in the following experiments. The algorithm uses a latent tree model to represent the cross-attribute correlations in the original dataset and perturbs the tree parameters via a distributed Laplace protocol to achieve DP guarantee for *each* local dataset. Additionally, a tree index based method TICQ can also be used to determine the minimum set of latent attribute pairs for constructing the latent tree, which helps to reduce the noise scale. The total privacy budget is consumed by three parts, namely the generation of latent attributes, quantification of latent attributes' correlations, and privatization of the tree parameters. For each dataset, we respectively compare the synthetic data utility of using the DPLT algorithm (referred to as DPLT) as well as the improved TICQ-DPLT algorithm (referred to as DPLT+). Moreover, we also present the utility of synthetic data generated under the non-private setting as a reference.

6.1.3 Parameter Configurations. In the following experiments, we conduct the collaborative training process for $T = 1500$ rounds. During local training, each local discriminator is updated for $T_d = 10$ steps with a batch size of $B = 1000$. For both the generator and discriminator, we use the `RMSprop` optimizer with a default learning rate of $\eta = 0.001$. Moreover, we apply the gradient perturbation when training the local discriminators, where the L_2-clip bound C is set to 1 and the noise scale σ varies according to the target privacy budget. We choose a different privacy budget $\epsilon \in \{0.5, 1, 2, 4, 8\}$ and $\delta = 10^{-5}$ so as to explore the influence of privacy on the framework performance. The ϵ here follows the traditional DP definition (Definition 1).

6.1.4 Evaluation Metrics. We evaluate the performance of our VertiGAN framework from two perspectives, namely the *utility evaluation* and the *privacy evaluation*. For the *utility evaluation*, we first compare the statistical similarity of synthetic data and real data. Then, we apply commonly-used machine learning models to investigate the utility of synthetic data in AI training tasks. For the *privacy evaluation*, we investigate the capability of our framework against membership inference attacks, where an attacker aims to use the synthetic dataset to determine whether a target record is used for training the GAN model.

6.1.5 Computation Environments. We perform all the experiments on a NVIDIA Quadro RTX 6000 GPU. In Table 3, we compare the training time (sec) of our VertiGAN framework and the baseline DPLT+ algorithm regarding all the datasets.

Table 3: Computation time (sec) of the proposed VertiGAN framework and baseline DPLT+ algorithm regarding different datasets. For VertiGAN, we perform 1500 global rounds and report the total training time.

Dataset	Web	Vehicle	Census	Twitter	HAR	Dilbert
DPLT+	1516.20	1116.81	1341.47	3954.03	19598.39	34413.66
VertiGAN	485.35	396.35	424.42	510.26	3296.08	8387.30

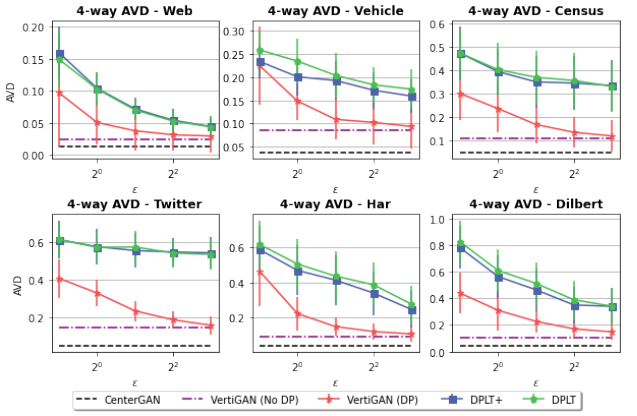

Figure 4: Average total variation distance (AVD) of four-way joint distributions between the real and synthetic data with respect to different privacy levels.

6.2 Utility: Statistical Similarity

We start our evaluation under the two-party setting, which is commonly used in existing VFL frameworks. Here, each party holds half of the attributes. We first evaluate the performance of VertiGAN by investigating whether the generated synthetic data can preserve similar statistical properties as real data. To this end, we respectively compare the k-way joint distributions and cross-attribute correlations of the real data and synthetic data and analyze their statistical similarity.

6.2.1 Comparison of Joint Distributions. For the analysis of joint distributions, we used the Average Variant Distance (AVD) to quantify the distribution difference between the real data and synthetic data, as used in [53], which is defined as

$$AVD = \frac{1}{2} \sum_{\omega \in \Omega} |P_{real}(\omega) - P_{syn}(\omega)|, \tag{19}$$

where Ω is the domain of all the k-way attribute combinations, ω is one of the combinations, $P_{real}(\omega)$ and $P_{syn}(\omega)$ are joint distributions of real and synthetic data. More specifically, assume the attribute combination ω has a domain size of $|\omega|$, P_{real} and P_{syn} are $|\omega|$-dimensional vectors, where each entry is the probability of a specific value combination (namely the ratio of occurrence in the entire real or synthetic dataset). For each dataset, we randomly chose 100 k-way attribute combinations and compute the average distribution difference.

AVD Regarding the Privacy Budget ϵ. In Figure 4, we first compare the four-way AVD of the synthetic data generated by the

VertiGAN framework as well as the two baseline algorithms under different privacy levels. We also report the results under the fully-centralized setting and the results of the proposed framework under the non-private setting as a reference. The error bars represent the 95% confidence interval (also for the remaining experimental results). It can be seen that the AVD of all the algorithms reduces with the increase of ϵ. Nonetheless, for all the datasets, the synthetic data generated by the VertiGAN framework consistently achieve a smaller AVD in comparison with the baseline methods, which indicates a better capability of our VertiGAN framework in capturing the multivariate distributions. Moreover, there is a more distinctive gap in AVD between the baseline algorithms and VertiGAN for the datasets with a larger domain size. It can be observed that when $\epsilon \geq 4$, the AVD of the baseline algorithms is almost two to three times in comparison with VertiGAN. This is because a larger domain size refers to more cross-attribute combinations. Since the baseline algorithms are supposed to evenly split the privacy budget to all the attribute pairs, the increase in domain size may cause each attribute pair being allocated with an insufficient privacy budget, which may result in serious degradation of data utility. In comparison, VertiGAN applies DP perturbation to the discriminator's gradients and is not directly related to the domain size. Therefore, the increase of domain size does not significantly affect the utility of the synthetic data generated by VertiGAN.

AVD Regarding the Multivariate Dimension k. We further analyze the AVD with varied multivariate dimension k to gain a deeper insight into VertiGAN's capability in the context of complex datasets. To this end, we choose $k \in \{2, 3, 4, 5, 6\}$ and compare the k-way AVD of using VertiGAN as well as the baseline algorithms. We present the results under $\epsilon = 2$ in Figure 5. Similarly, we also report the k-way AVD under the centralized setting and under the non-private VertiGAN setting as a reference. It can be seen that for all the datasets, VertiGAN steadily shows a smaller k-way AVD compared to the baseline algorithms. Moreover, although the baseline algorithms achieve similar AVD when k is small, the difference gets distinctively larger with an increase of k. Especially, for all the datasets, the 5-way and 6-way AVD of the baseline algorithms are almost twice that of VertiGAN. This indicates that our framework is more adept at capturing the information of high-dimensional joint distributions of real data.

6.2.2 Comparison of Correlation. We further visualize the correlation coefficient matrix of real data and synthetic data with heat maps in order to better understand the capability of our method in capturing and preserving the cross-attribute correlations. Figure 6 shows the comparison result of the different datasets with $\epsilon = 8$. For each dataset, we respectively select 10 attributes from each party and present the correlation matrix of the 20 attributes. From the visualization results, it can be seen that the correlation of synthetic data is similar to the correlation of real data, which further demonstrates that the synthetic data successfully preserves the attribute correlations of real data.

6.3 Utility: AI Training Performance

Next, we investigate the utility of synthetic data in AI training tasks. To this end, we train two classification models M_{real} and

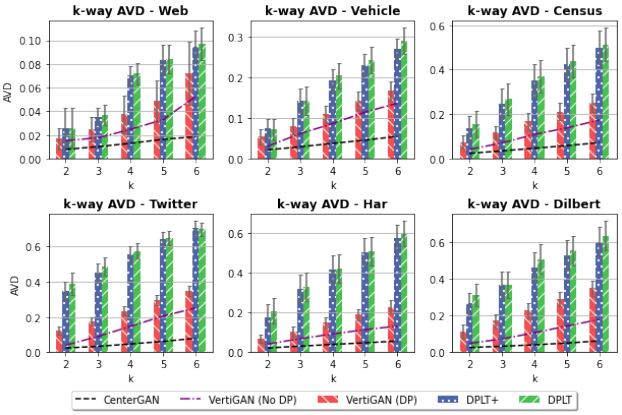

Figure 5: Average total variation distance (AVD) of k-way joint distributions between the real and synthetic data with respect to different dimensions of the joint distribution.

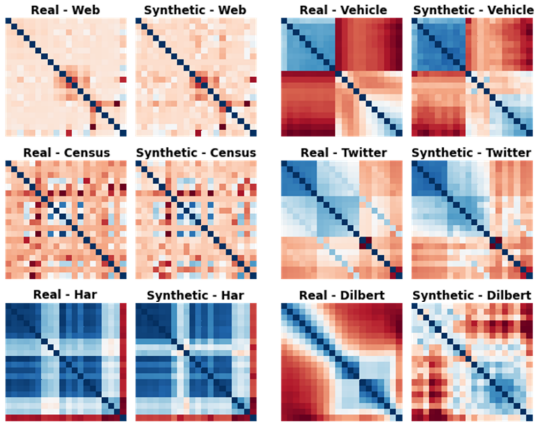

Figure 6: Correlation comparison between the real and synthetic data with $\epsilon = 8$. For each dataset, we present the correlations of 20 attributes, where each party contributes 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.

M_{syn}, respectively, with real data and synthetic data. Then, we test both models with an amount of held-out real data and compare the test accuracy, namely, Acc_{real} and Acc_{syn}. Intuitively, if Acc_{syn} is close to Acc_{real}, we consider the synthetic data to be of high utility which can replace real data for AI training tasks.

In the experiments, we use the Multi-layer Perceptron (MLP) classifier as the target AI model. We train both M_{real} and M_{syn} ten times and compute the averaged Acc_{real} and Acc_{syn}. In Table 4, we present the accuracy of the MLP classifiers evaluated on different datasets. For each dataset, we compare the Acc_{syn} of synthetic data generated under the non-private centralized and VertiGAN setting, as well as that generated by the private DPLT and VertiGAN frameworks with $\epsilon \in \{0.5, 2, 8\}$. It can be observed that although all the algorithms show a higher Acc_{syn} with an increase of ϵ, the accuracy of VertiGAN is generally higher than the baselines for

Table 4: Classification accuracy of MLP models evaluated on synthetic data generated under different privacy settings.

Dataset	Acc. Real	Acc. Synthetic (No DP)		Acc. Synthetic (With DP)			
		Center GAN	Verti GAN	ϵ	DPLT	DPLT+	Verti GAN
Web	0.8453	0.8276	0.8079	0.5	0.7114	0.7124	0.6970
				2	0.7251	0.7238	0.7776
				8	0.7385	0.7484	0.7900
Vehicle	0.8204	0.8074	0.7984	0.5	0.7385	0.7208	0.7498
				2	0.7596	0.7373	0.7627
				8	0.7690	0.7729	0.7840
Census	0.9858	0.9820	0.9732	0.5	0.8822	0.8870	0.9088
				2	0.9027	0.9092	0.9432
				8	0.9465	0.9487	0.9655
Twitter	0.8209	0.8180	0.7871	0.5	0.7277	0.7274	0.7445
				2	0.7371	0.7397	0.7701
				8	0.7520	0.7580	0.7822
HAR	0.9532	0.8990	0.8414	0.5	0.4228	0.4570	0.5368
				2	0.5519	0.5702	0.7022
				8	0.6038	0.6284	0.7746
Dilbert	0.9394	0.8651	0.8010	0.5	0.2722	0.2988	0.5525
				2	0.4331	0.4353	0.6241
				8	0.5434	0.5672	0.7134

all privacy levels, especially for complex datasets. In particular, with $\epsilon = 8$, the synthetic data generated by VertiGAN achieves around 2% $\sim$ 15% improvement of Acc_{syn} compared to the baseline algorithms. The results indicate that our framework has a better capacity for preserving the hidden patterns and correlations of real data compared to the baselines. The generated synthetic data can be effectively used for data mining and AI training tasks.

6.4 Ablation Study

We further conduct a series of ablation studies to investigate how the size of local datasets, the imbalanced splitting of attribute sets, and the increase of local parties impact the performance of the VertiGAN framework and synthetic data utility.

6.4.1 Impact of the Number of Records. To start with, in the previous experiments, we assume that the local parties share data of 10^5 records. We further investigate how varying the number of local records affects the framework's performance. To this end, we respectively vary the size of local datasets with 10^4, 10^5, and 10^6 records and conduct experiments under different privacy levels. In Figure 7, we present the 4-way AVD of the **Vehicle**, **Census**, and **HAR** datasets with $\epsilon = \{0.5, 2, 8\}$. It can be seen that using a larger number of records can significantly improve the data utility, especially in high-privacy regimes. For instance, when $\epsilon = 0.5$, for all the datasets, the AVD with 10^4 records is 2 $\sim$ 4 times the results with 10^5 records. This is because the privacy loss of each iteration is related to the *sampling rate* γ, as shown in Lemma 1. Therefore, with a fixed batch size of B, increasing the total number of records

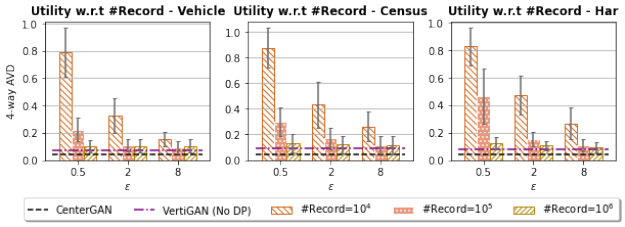

Figure 7: 4-way AVD under the two-party settings with 10^4, 10^5, and 10^6 records under the privacy level $\epsilon = \{0.5, 2, 8\}$.

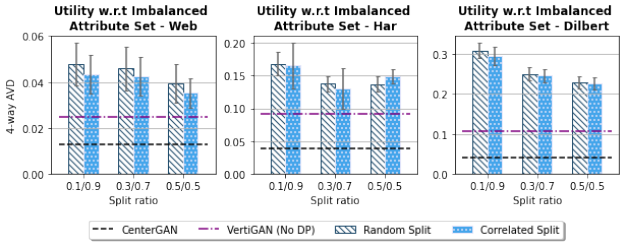

Figure 8: 4-way AVD under the two-party settings with $\epsilon = 8$ and attribute split ratio from $\{0.1/0.9, 0.3/0.7, 0.5/0.5\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly-correlated attributes are assigned to one of the parties.

leads to a decrease in privacy loss. In other words, the framework only needs to add a smaller amount of noise to achieve the same privacy level, which largely enhances the utility of synthetic data. On the other hand, for $\epsilon = 8$, the AVD with 10^6 is similar to the results with 10^5 records. This is because larger privacy budgets result in less noise being injected during training, hence the model can already converge well with 10^5 records. In this case, using larger datasets offers a comparatively smaller contribution to the utility.

6.4.2 Impact of Imbalanced Attribute Sets. Next, in addition to exploiting the setting where the entire attribute set is evenly split and held by two local parties, we also investigate whether the utility of the synthetic data will differ if the local parties possess an imbalanced number of attributes. To this end, we split the entire attribute set with a ratio of 0.1/0.9, 0.3/0.7, and 0.5/0.5 (i.e., an even split) and compare the data utility under different privacy levels. Moreover, we also explore whether the imbalanced split of strongly-correlated attributes affects the data utility. To this end, we first compute the pair-wise correlation of all the attributes and apply hierarchical clustering to group the most correlated attributes. Then, we construct the imbalanced attribute sets in two ways: *random split* and *correlated split*. The former randomly splits the attribute set according to the split ratio, while the latter manually assigns the strongly correlated attributes to one of the local parties. We conduct experiments under different split ratios following both split fashions and report the results in Figure 8. It can be observed that an imbalanced attribute set can lead to a degradation of framework performance. In contrast, assigning the strongly-correlated attributes to one of the parties slightly improves the data utility

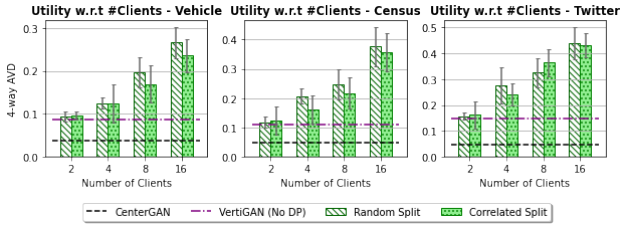

Figure 9: 4-way AVD under the two-party settings with $\epsilon = 8$ and the number of clients from $\{2, 4, 8, 16\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly-correlated attributes are assigned to a subset of the parties.

compared to the random setting. Intuitively, when the attributes belong to different generator branches, the framework may suffer from a certain information loss on the pair-wise correlations. In contrast, the correlation information can be better preserved when both attributes belong to the same branch, hence leading to higher data utility.

6.4.3 Impact of the Number of Local Parties. Besides the impact of imbalanced splitting, we also analyze the effects of varying the number of local parties on the framework performance. To this end, we respectively perform the data publication process using the different methods under the settings consisting of 2, 4, 8, and 16 local parties and compare the utility of synthetic data. In Figure 9, we present the 4-way AVD with different numbers of local parties under $\epsilon = 8$. It can be observed that the framework performance degrades with the increase of local parties. This might be because the joint distributions and correlations are more difficult to be captured when the correlated attributes are spread over multiple local parties. On the other hand, similar to observations in Section 6.4.2, when assigning all the strongly-correlated attributes to a subset of local parties (*i.e.*, by using *correlated splitting*), the cross-attribute correlations can be better preserved and the data utility can be further improved.

6.5 Empirical Privacy Analysis

Although choosing a larger privacy budget ϵ can distinctively improve the data utility, this may lead to increased privacy leakage. In order to obtain a better understanding of the utility-privacy trade-off, we conduct a membership inference attack to empirically analyze the privacy protection capabilities of our framework under different privacy settings. We follow the black-box MIA protocol proposed in [24], which uses the distance of a target record to the synthetic dataset to infer the membership information. The intuition is that the generator tends to generate synthetic data close to the training data. Therefore, given a target record x, let $U_\tau(x) = \{x' | d(x, x') \leq \tau\}$ denote the τ-neighborhood of x with respect to the distance metric d. Then, we randomly generate a synthetic dataset $\mathcal{X}_{syn}$ with n records and compute the ratio that the synthetic records fall into the neighborhood of x, namely

$$\hat{f}_\tau(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[x^i_{syn} \in U_\tau(x)], \tag{20}$$

Table 5: MIA accuracy under different privacy settings.

	CenterGAN	VertiGAN			
	No DP	No DP	ϵ=8	ϵ=2	ϵ=0.5
Vehicle	0.5841	0.5758	0.5538	0.5448	0.5287
Web	0.6008	0.5844	0.5633	0.5367	0.5223
Census	0.6509	0.6394	0.6171	0.5800	0.5421
Twitter	0.6746	0.6672	0.6320	0.5980	0.5676
HAR	0.5784	0.5637	0.5324	0.5241	0.5160
Dilbert	0.6044	0.5856	0.5623	0.5433	0.5386

where x^i_{syn} is the i^{th} synthetic record. Obviously, the higher the $\hat{f}_\tau(x)$, the more likely it is that x is included in the training data.

In our experiments, we construct the target dataset by randomly sampling 100 training records (denoted as $\mathcal{X}_{in}$) and 100 testing records (denoted as $\mathcal{X}_{out}$). Then, we generate a synthetic dataset $\mathcal{X}_{syn}$ with 10^4 records and use the normalized Hamming distance to measure the minimum distance between each target record and the synthetic data. Following [24], we set τ as the median of the minimum distance of each record. Given the ground truth label and the predicted membership probability, we compute the averaged attack accuracy under different privacy settings. The results are reported in Table 5. It can be observed that synthetic data generated by non-private GANs are still likely to reveal the membership information of the target record. In particular, for **Twitter** and **Census** dataset, the attack accuracy under the non-private setting is more than 65%. On the other hand, applying DP to our VertiGAN framework can effectively reduce attack accuracy. With $\epsilon = 8$, the attack accuracy is reduced by 2% $\sim$ 4%, while with $\epsilon = 0.5$, the attack accuracy is reduced by 5% $\sim$ 10%. The results demonstrate that our framework is able to mitigate the risk of membership inference attacks and can provide strengthened privacy protection to the local data.

7 DISCUSSIONS AND FUTURE WORK

In this section, we discuss potential extensions of our framework, current limitations, and directions for future work.

7.1 Extension to Other Data Types

In Section 6, we demonstrated that the VertiGAN framework is effective in publishing vertically-partitioned categorical datasets and achieves better data utility compared to previous statistics-based baselines. Moreover, our framework can be further extended to more complex settings where each party holds different types of data. For instance, in a healthcare scenario, a group of hospitals can use the framework to publish a joint dataset containing patients' CT images and physical symptoms for future medical research. This can be realized by modifying the structure of the models and using the advanced layers. For instance, we can respectively adopt convolution layers and recurrent layers to enhance the feature extraction on image data and time-series data. Despite the variation of the layers and model structures, the main workflow of the VertiGAN framework remains unchanged. In Figure 10, we further demonstrate the framework's feasibility in the context of image data. Here, we assume that there are three local parties respectively

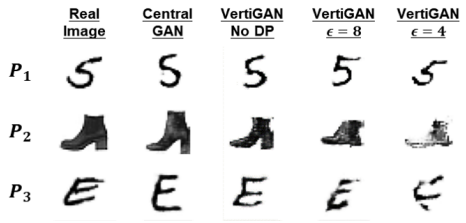

Figure 10: Results of image data synthesis under a three-party setting. Each row represents the synthetic images generated by one local party under different privacy settings.

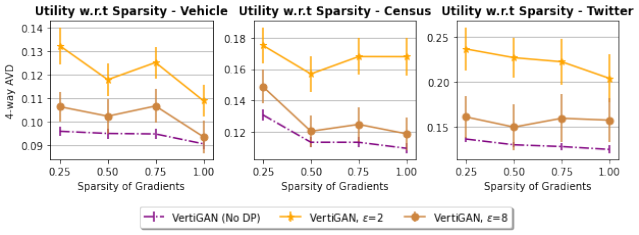

Figure 11: Four-way AVD between the real and synthetic data with respect to different gradient sparsity ratios.

holding handwritten digits from **MNIST**, handwritten letters from **Extended MNIST**, and article images from **Fashion-MNIST**. We construct a global generator with three output branches and the corresponding local discriminators and analyze the quality of synthetic images generated under different privacy settings. Note that the synthetic data are randomly generated and hence are not identical to the real data. Nevertheless, it can still be observed that our framework is capable of jointly synthesizing all three categories of images of different local clients, and the generated data enjoys a satisfactory level of quality under a larger privacy budget.

7.2 Reduction of Communication Cost

As described in Section 5.4, in each global round, the parameters and gradients of the global generator are repeatedly exchanged between the server and local parties. This may result in a high communication cost, especially for high-dimensional models. One possible approach for mitigating the upload communication cost is to process the generator's gradients with top-k sparsification and send the sparsified gradients to the server. In Figure 11, we investigate how the sparsification level affects the utility of synthetic data. Here, we choose the top-k ratio from $\{0.25, 0.5, 0.75, 1\}$ and compare the corresponding 4-way AVD of the synthetic data under the privacy level of $\epsilon \in \{2, 8\}$. It can be observed that even processing the gradients with a top-k ratio of 0.25 can still achieve data utility comparable to returning the entire gradients. The results demonstrate the effectiveness of gradient sparsification in reducing the upload communication cost. On the other hand, a few recent studies also proposed to use dropout [8] and model pruning [30] to reduce the size of the broadcast global model. Our framework can be further improved following this idea: before the training starts, the server broadcasts the initialized global generator to all the local parties. Then, during training, instead of broadcasting the entire global generator, the server only sends the parameters of the common layers G^0 and the corresponding branch G^i to the party $\mathcal{P}^i$, which is enough for $\mathcal{P}^i$ to produce the synthetic data $\tilde{X}^i = G^i(G^0)(Z)$ on the local side (see Section 5.2). The improvement not only reduces the *download* communication but also prevents the local parties from inferring the inputs of the other parties.

7.3 Protection for the Uploaded Gradients

In this paper, we apply DP perturbation to the discriminator and enforce privacy guarantees to the generator according to the post-processing property. Nevertheless, even though the global generator is not directly trained on the local data, the gradients derived by the local discriminators may still reveal sensitive information about local data. Considering that recent studies in FL [6, 29] adopt SMC or local differential privacy (LDP) for encrypting or perturbing local updates, such protection techniques may also be applicable to our framework. For instance, we can use SMC protocols to encrypt the real gradients on the local side before sending them to the server. In this way, the server cannot obtain the individual real gradients but only the sum of all the gradients after the decryption. However, the use of SMC protocols may increase the communication and computational cost of the framework due to the key generation and exchange process. On the other hand, LDP-based solutions add random noise to the local gradients, which will not largely affect efficiency. Nevertheless, it may cause significant utility loss due to the limited number of local parties under the vertical setting. Hence, how to protect the uploaded gradients regarding security, utility, and efficiency will be an important direction for future work.

8 CONCLUSION

Due to the great variety in service scenarios, user data in real-life applications are often vertically partitioned and distributed among different local parties. Although it is of great benefit for data analysts to explore the hidden correlations of attributes of all the local parties, publishing the vertically-partitioned data raises both privacy and utility concerns.

In this paper, we follow the idea of synthetic data generation and propose VertiGAN, the first GAN-based framework for privately publishing vertically-partitioned data. Different from the prior statistics-based solutions, our framework adopts a distributed GAN architecture, where a global generator is adversarially trained with a group of local discriminators to learn the distribution of all parties' local data and used to directly generate synthetic integrated data on the server side. Moreover, we apply DP perturbation during the training process to ensure strict privacy guarantees for the local data. Experimental evaluation with real-world datasets shows that our framework significantly outperforms the statistics-based baseline algorithms for publishing high-dimensional vertically-partitioned data. The synthetic data generated by our framework preserves very similar statistical properties as real data and can replace real data for data mining and model training tasks.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments for improving this paper.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Vienna, Austria, 308–318.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *21st European Symposium on Artificial Neural Networks*. Ciaco-i6doc.com, Bruges, Belgium, 437–442.

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *34th International Conference on Machine Learning*, Vol. 70. PMLR, Sydney, NSW, Australia, 214–223.

[4] Ching-Yuan Bai, Hsuan-Tien Lin, Colin Raffel, and Wendy Chi-wen Kan. 2021. On Training Sample Memorization: Lessons from Benchmarking Generative Modeling with a Large-scale Competition. In *27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Virtual Event, Singapore, August 14-18, 2021, 2534–2542.

[5] Gabrielle Berman, Sara de la Rosa, and Tanya Accone. 2018. Ethical Considerations When Using Geospatial Technologies for Evidence Generation. *Innocenti Discussion Papers* 2018-02 (2018).

[6] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Dallas, TX, USA, 1175–1191.

[7] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. 2011. "You Might Also Like:" Privacy Risks of Collaborative Filtering. In *2011 IEEE Symposium on Security and Privacy*. IEEE, Berkeley, CA, USA, 231–246.

[8] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. *CoRR* abs/1812.07210 (2018).

[9] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium*. USENIX Association, Santa Clara, CA, USA, 267–284.

[10] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, USA, 343–362.

[11] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. 2020. VAFL: A Method of Vertical Asynchronous Federated Learning. *CoRR* abs/2007.06081 (2020).

[12] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. 2021. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intelligent Systems* 36, 6 (2021), 87–98.

[13] Michele Ciampi and Claudio Orlandi. 2018. Combining Private Set-Intersection with Secure Two-Party Computation. In *11th International Conference on Security and Cryptography for Networks*, Vol. 11035. Springer, Amalfi, Italy, 464–482.

[14] Ivan Damgård and Mads Jurik. 2001. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *4th International Workshop on Practice and Theory in Public Key Cryptography*, Vol. 1992. Springer, Cheju Island, Korea, 119–136.

[15] Sumit Kumar Debnath and Ratna Dutta. 2015. Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter. In *18th Information Security Conference*, Vol. 9290. Springer, Trondheim, Norway, 209–226.

[16] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

[17] Marco F. Duarte and Yu Hen Hu. 2004. Vehicle Classification in Distributed Sensor Networks. *Journal of Parallel and Distributed Computing* 64, 7 (2004), 826–838.

[18] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[19] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X. Liu, and Ting Wang. 2022. Label Inference Attacks Against Vertical Federated Learning. In *31st USENIX Security Symposium*. USENIX Association, Boston, MA, 1397–1414.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *27th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Montreal, Quebec, Canada, 2672–2680.

[21] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander R. Statnikov, Wei-Wei Tu, and Evelyne Viegas. 2019. Analysis of the AutoML Challenge Series 2015-2018. In *Automated Machine Learning - Methods, Systems, Challenges*. Springer, Berlin, Germany, 177–219.

[22] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private Federated Learning on Vertically Partitioned Data via Entity Resolution and Additively Homomorphic Encryption. *CoRR* abs/1711.10677 (2017).

[23] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. LOGAN: Membership Inference Attacks Against Generative Models. *Proceedings on Privacy Enhancing Technologies* 2019, 1 (2019), 133–152.

[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, 4 (2019), 232–249.

[25] Yan Huang, David Evans, and Jonathan Katz. 2012. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?. In *19th Annual Network and Distributed System Security Symposium*. The Internet Society, San Diego, California, USA.

[26] Wei Jiang and Chris Clifton. 2006. A Secure Distributed Framework for Achieving k-Anonymity. *The International Journal on Very Large Data Bases* 15, 4 (2006), 316–333.

[27] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Comprehensive Analysis of Privacy Leakage in Vertical Federated Learning During Prediction. *Proceedings on Privacy Enhancing Technologies* 2022, 2 (2022), 263–281.

[28] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder. *Proceedings on Privacy Enhancing Technologies* 2022, 1 (2022), 481–500.

[29] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection. *ACM Transactions on Intelligent Systems and Technology* 13, 5, Article 74 (2022), 22 pages.

[30] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. 2022. Model Pruning Enables Efficient Federated Learning on Edge Devices. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–13.

[31] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *7th International Conference on Learning Representations*. OpenReview.net, New Orleans, LA, USA.

[32] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[33] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations*. OpenReview.net, Vancouver, BC, Canada.

[34] François Kawala, Ahlame Douzal-Chouakria, Eric Gaussier, and Eustache Dimert. 2013. Prédictions d'Activité dans les Réseaux Sociaux en Ligne. In *4ième Conférence sur les Modèles et l'Analyse des Réseaux: Approches Mathématiques et Informatiques*. France.

[35] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*. OpenReview.net, Banff, AB, Canada.

[36] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, and Klaus A Kuhn. 2014. A Flexible Approach to Distributed Data Anonymization. *Journal of Biomedical Informatics* 50 (2014), 62–76.

[37] Yann LeCun, Fu Jie Huang, and Léon Bottou. 2004. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, Washington, DC, USA, 97–104.

[38] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially Private Synthesization of Multi-dimensional Data Using Copula Functions. In *Proceedings of the 17th International Conference on Extending Database Technology*, Vol. 2014. OpenProceedings.org, Athens, Greece, 475–486.

[39] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2022. Label Leakage and Protection in Two-party Split Learning. In *Tenth International Conference on Learning Representations*. OpenReview.net, Virtual Event.

[40] Yang Liu, Zhihao Yi, and Tianjian Chen. 2020. Backdoor Attacks and Defenses in Feature-partitioned Collaborative Learning. *CoRR* abs/2007.03608 (2020).

[41] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature Inference Attack on Model Predictions in Vertical Federated Learning. In *37th IEEE International Conference on Data Engineering*. IEEE, Chania, Greece, 181–192.

[42] H. Brendan McMahan and Galen Andrew. 2018. A General Approach to Adding Differential Privacy to Iterative Training Procedures. *CoRR* abs/1812.06210 (2018).

[43] Ilya Mironov. 2017. Rényi Differential Privacy. In *30th IEEE Computer Security Foundations Symposium*. IEEE, Santa Barbara, CA, USA, 263–275.

[44] Noman Mohammed, Dima Alhadidi, Benjamin CM Fung, and Mourad Debbabi. 2013. Secure Two-party Differentially Private Data Release for Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 11, 1 (2013), 59–71.

[45] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *5th International Conference on Learning Representations*. OpenReview.net, Toulon, France.

[46] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis Based on Generative Adversarial Networks. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1071–1083.

[47] John Platt. 1998. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Technical Report MSR-TR-98-14. Microsoft.

[48] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A. Hoeh. 2021. PyVertical: A Vertical Federated Learning Framework for Multi-headed SplitNN. *CoRR* abs/2104.00489 (2021).

[49] 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*. IEEE, San Jose, CA, USA, 3–18.

[50] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic Data – Anonymisation Groundhog Day. In *31st USENIX Security Symposium*. ACM, Boston, MA, USA, 1451–1468.

[51] Yan Sun, Lihua Yin, Licai Liu, and Shuang Xin. 2014. Toward Inference Attacks for k-Anonymity. *Personal and Ubiquitous Computing* 18, 8 (2014), 1871–1880.

[52] Latanya Sweeney. 2002. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 5 (2002), 557–570.

[53] Peng Tang, Xiang Cheng, Sen Su, Rui Chen, and Huaxi Shao. 2019. Differentially Private Publication of Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 18, 2 (2019), 780–795.

[54] Aleksei Triastcyn and Boi Faltings. 2020. Federated Generative Privacy. *IEEE Intelligent Systems* 35, 4 (2020), 50–57.

[55] Jaideep Vaidya and Chris Clifton. 2004. Privacy Preserving Naïve Bayes Classifier for Vertically Partitioned Data. In *Fourth SIAM International Conference on Data Mining*. SIAM, Lake Buena Vista, Florida, USA, 522–526.

[56] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and Scott Patterson. 2008. Privacy-preserving Decision Trees Over Vertically Partitioned Data. *ACM Transactions on Knowledge Discovery from Data* 2, 3 (2008), 1–27.

[57] Gerrit van den Burg and Chris Williams. 2021. On Memorization in Probabilistic Deep Generative Models. In *Annual Conference on Neural Information Processing Systems 2021*. OpenReview.net, Virtual, 27916–27928.

[58] Boxin Wang, Fan Wu, Yunhui Long, Luka Rimanic, Ce Zhang, and Bo Li. 2021. DataLens: Scalable Privacy Preserving Training via Gradient Compression and Aggregation. In *2021 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, Republic of Korea, 2146–2168.

[59] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. 2020. Hybrid Differentially Private Federated Learning on Vertically Partitioned Data. *CoRR* abs/2009.02763 (2020).

[60] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled Renyi Differential Privacy and Analytical Moments Accountant. In *22nd International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 89)*. PMLR, Naha, Okinawa, Japan, 1226–1235.

[61] 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. In *6th International Conference on Learning Representations*. OpenReview.net, Vancouver, BC, Canada.

[62] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. 2007. Minimality Attack in Privacy Preserving Data Publishing. In *33rd International Conference on Very Large Data Bases*. ACM, Vienna, Austria, 543–554.

[63] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy Preserving Vertical Federated Learning for Tree-based Models. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2090–2103.

[64] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018).

[65] Kai Yang, Tao Fan, Tianjian Chen, Yuanming Shi, and Qiang Yang. 2019. A Quasi-Newton Method Based Vertical Federated Learning Framework for Logistic Regression. *CoRR* abs/1912.00513 (2019).

[66] Andrew Chi-Chih Yao. 1982. Protocols for Secure Computations (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science*. IEEE, Chicago, Illinois, USA, 160–164.

[67] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In *32th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., Vancouver, BC, Canada, 5509–5519.

[68] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Transactions on Database Systems* 42, 4 (2017), 25:1–25:41.

[69] Longling Zhang, Bochen Shen, Ahmed Barnawi, Shan Xi, Neeraj Kumar, and Yi Wu. 2021. FedDPGAN: Federated Differentially Private Generative Adversarial Networks Framework for the Detection of COVID-19 Pneumonia. *Information Systems Frontiers* 23, 6 (2021), 1403–1415.

[70] Nevin L Zhang. 2004. Hierarchical Latent Class Models for Cluster Analysis. *The Journal of Machine Learning Research* 5 (2004), 697–723.

[71] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially Private Releasing via Deep Generative Model. *CoRR* abs/1801.01594 (2018).

[72] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. 2022. Property Inference Attacks Against GANs. In *29th Annual Network and Distributed System Security Symposium*. The Internet Society, San Diego, CA, USA.

[73] Tianyuan Zou, Yang Liu, Yan Kang, Wenhan Liu, Yuanqin He, Zhihao Yi, Qiang Yang, and Ya-Qin Zhang. 2022. Defending Batch-Level Label Inference and Replacement Attacks in Vertical Federated Learning. *IEEE Transactions on Big Data* 1 (2022), 1–12.