VFLGAN-TS: Vertical Federated Learning-based Generative Adversarial Networks for Publication of Vertically Partitioned Time-series Data

YUAN XUN, National University of Singapore, Singapore, Singapore
ZILONG ZHAO, National University of Singapore, Singapore, Singapore
JIAYU LI, National University of Singapore, Singapore, Singapore
PROSANTA GOPE, Computer Science, The University of Sheffield, Sheffield, United Kingdom of Great Britain and Northern Ireland
BIPLAB SIKDAR, National University of Singapore, Singapore, Singapore

In the current artificial intelligence (AI) era, the scale and quality of the dataset play a crucial role in training a high-quality AI model. However, often original data cannot be shared due to privacy concerns and regulations. A potential solution is to release a synthetic dataset with a similar distribution to the private dataset. Nevertheless, in some scenarios, the attributes required to train an AI model are distributed among different parties, and the parties cannot share the local data for synthetic data construction due to privacy regulations. In PETS 2024, we recently introduced the *first* Vertical Federated Learning-based Generative Adversarial Network (VFLGAN) for publishing vertically partitioned static data. However, VFLGAN cannot effectively handle time-series data, which contains both temporal and attribute dimensions. In this paper, we proposed VFLGAN-TS, which combines the ideas of attribute discriminator and vertical federated learning to generate synthetic time-series data in the vertically partitioned scenario. The performance of VFLGAN-TS is close to that of its centralized counterpart, which represents the upper limit for VFLGAN-TS. To further protect privacy, we apply a Gaussian mechanism to make VFLGAN-TS satisfy an (ϵ, δ)-differential privacy. Besides, we develop an enhanced privacy auditing scheme to evaluate the potential privacy breach through the framework of VFLGAN-TS and synthetic datasets.

CCS Concepts: • **Security and Privacy Applications**;

Additional Key Words and Phrases: Generative adversarial networks, Vertical federated learning, Privacy-preserving data publication, Time-series data generation

1 Introduction

The performance of deep-learning (DL) models is closely linked to the quality and scale of the training dataset. For example, key advancements in image perception [44], language understanding [44], recommendation systems [29], and anomaly detection [49] are attributed to high-quality datasets like ImageNet [10], expansive textual dataset [9], Netflix rating dataset [3], and IOT dataset [49], respectively. However, a significant challenge is the vertically partitioned scenario, where separate entities hold different attributes required for the construction of the dataset. For instance, a bank may have a client's financial data, while health records are maintained by

Authors' Contact Information: Yuan Xun, National University of Singapore, Singapore, Singapore; e-mail: e0919068@u.nus.edu; Zilong Zhao, National University of Singapore, Singapore, Singapore; e-mail: z.zhao@nus.edu.sg; Jiayu Li, National University of Singapore, Singapore, Singapore; e-mail: li.jiayu@nus.edu.sg; Prosanta Gope, Computer Science, The University of Sheffield, Sheffield, United Kingdom of Great Britain and Northern Ireland; e-mail: p.gope@sheffield.ac.uk; Biplab Sikdar, National University of Singapore, Singapore, Singapore; e-mail: bsikdar@nus.edu.sg.

ACM 2471-2574/2025/11-ART
https://doi.org/10.1145/3776587

hospitals or insurers. Integrating these diverse attributes can offer a more comprehensive understanding of customers, thereby enhancing decision-making, as highlighted in several studies [22, 43].

Although integrating dispersed data attributes offers clear benefits, practical implementation faces obstacles due to privacy concerns and stringent data protection regulations like GDPR [47] that prohibit sharing the original data among different institutions. A potential solution to this challenge is to publish a synthetic dataset that reflects the private dataset's distribution without revealing any raw data. Generating synthetic datasets usually requires access to all the attributes of the private datasets. However, in the vertically partitioned scenario, each party holds part of the attributes and cannot share its local attributes. This problem is referred to as the publication of vertically partitioned data. The objective of this project is to release a synthetic dataset that mimics the distribution of the private dataset whose attributes are owned by different parties and cannot be shared among the parties.

In our previous study [60], we introduced the first Vertical Federated Learning-based Generative Adversarial Networks (VFLGAN) to publish vertically partitioned static data. However, VFLGAN struggles with generating time-series data due to its inability to simultaneously learn the correlation along the temporal and attribute dimensions, as shown in the experimental results in Section 5.2. To address this issue, we developed the first generative model for vertically partitioned time-series data, Vertical Federated Learning-based Generative Adversarial Networks for Time Series (VFLGAN-TS). VFLGAN-TS integrates the ideas of the attribute discriminator proposed in [39] and Vertical Federated Learning (VFL) [27]. The attribute discriminator effectively learns the temporal correlations, and VFL satisfies the privacy constraints of vertically partitioned scenarios and learns the attribute correlations.

As discussed in [41], synthetic datasets are not safe from membership inference attacks (MIA). To counteract such vulnerabilities, Differential Privacy (DP) [11] offers a promising privacy protection strategy. In this paper, we employ the Gaussian mechanism proposed in [60] to make VFLGAN-TS satisfy an (ϵ, δ)-DP to further protect privacy, and we name the differentially private version with DPVFLGAN-TS. On the other hand, although DP can offer worst-case privacy assurances [32], most real-world datasets do not contain the worst-case sample. Thus, in [60], we proposed a practical auditing scheme to evaluate the potential privacy breach through synthetic datasets. However, our test shows that the scheme proposed in [60] is not effective for time-series synthetic data (details shown in Table 7). In this paper, we enhance the auditing scheme to evaluate the potential privacy breach through the VFLGAN-TS framework and the generated synthetic time-series datasets. To summarize, our contributions can be summarized as follows:

- We introduce VFLGAN-TS, the *first* solution for the publication of vertically partitioned time-series data. Besides, we adapt the Gaussian mechanism proposed in [60] to equip VFLGAN-TS with (ϵ, δ)-DP.
- We enhance the auditing scheme proposed in [60] to evaluate potential privacy breaches through VFLGAN-TS framework and synthetic datasets.
- Extensive experiments were conducted to evaluate the quality and privacy breaches of synthetic datasets generated by VFLGAN-TS.

2 Related Work

In this section, we introduce related work about the publication of vertically partitioned data, Generative Adversarial Networks (GANs) for time-series data, differentially private mechanisms for GANs, and privacy auditing methods.

2.1 Publication of Vertically Partitioned Data

DistDiffGen [35] is a secure two-party algorithm using the exponential mechanism to achieve ϵ-DP. However, it is only suitable for classification tasks and lacks utility for other standard data analysis tasks [43]. The authors of [43] proposed DPLT, which also satisfies ϵ-DP but is limited to discrete datasets and evenly distributes the privacy budget across all attributes. As noted by [22], increased data dimensionality can exponentially increase the noise scale, leading to significant utility loss. The first GAN-based approach, VertiGAN [22], utilizes FedAvg [30] to meet privacy needs in vertically partitioned scenarios but struggles with learning attribute correlation. VFLGAN, integrating VFL and WGAN_GP [16], was proposed in [60] as the most effective method for vertically partitioned static data and the differentially private version satisfies (ϵ, δ)-DP. However, none of these methods is suitable for time-series data. In [56], the authors proposed a privacy-preserving data augmentation model for federated learning of heterogeneous data. However, it focuses on horizontally partitioned scenarios and image data generation rather than vertically partitioned scenarios and time-series data generation. In [54], the authors proposed a federated learning-based Domain Adaptation Method. In [55], the authors apply VFL for data augmentation, i.e., generating synthetic features given the raw data of local parties, which differs from the data generation studies presented in this paper. Besides, [55] focuses on image data, while this paper focuses on time-series data. [48] studies heterogeneous local models in the VFL framework, while this paper focuses on homogeneous local models in the VFL framework. [26] studies heterogeneous local models in the VFL framework, while this paper focuses on homogeneous local models in the VFL framework. [48] address non-aligned samples in practical VFL while we assume most samples are aligned in this paper.

2.2 GANs for Time-Series Data

In [14, 34, 37], the GAN framework has been directly applied to time-series data, but the effectiveness in capturing temporal correlations is limited [59]. To address this, TimeGAN [59] integrates supervised learning within the GAN framework to better capture temporal dynamics. Fourier Flows [2] employ a discrete Fourier transform (DFT) to convert time series into fixed-length spectral representations, followed by a chain of spectral filters leading to an exact likelihood optimization. GT-GAN [21] integrates various techniques, including GANs, neural ordinary/controlled differential equations, and continuous time-flow processes, into a single framework for time series synthesis. Still, its complexity makes adaptation to the VFL framework challenging. CosciGAN [39] advances in generating time series data by using channel discriminators to learn temporal distributions of each attribute and a central discriminator to understand attribute intercorrelations. CosciGAN outperforms both TimeGAN and Fourier Flows as shown in [39]. In this paper, we combine the channel/attribute discriminator and VFL framework to publish vertically partitioned time-series data.

2.3 Differentially Private Mechanisms for GANs

In [57, 61], the authors proposed variants of DPSGD [40] to train discriminators privately, while using non-private SGD to train generators. PATE-GAN [23] utilizes the PATE framework [36] to ensure differential privacy. It begins by training multiple non-private discriminators with non-overlapped subsets. These discriminators are then used to train a student discriminator that satisfies (ϵ, δ)-DP, which in turn trains the generator. Another approach, GS-WGAN [6], trains discriminators non-privately with non-overlapped subsets but sanitizes the backward gradients between discriminators and the generator using a Gaussian mechanism to achieve (ϵ, δ)-DP. However, as discussed in [60], the above methods are unsuitable for vertically partitioned scenarios. This paper adapts the Gaussian mechanism proposed in [60] to VFLGAN-TS, ensuring it has an (ϵ, δ)-DP. Blockchain [50, 51] is another important method to protect users' privacy. However, it suffers from extremely huge computational requirements compared with DP.

2.4 Privacy Auditing Methods

Privacy auditing encompasses two primary research directions. The first involves estimating the lower bounds of ϵ for ϵ-DP using poisoning samples, as explored in [20, 25, 28, 42]. However, these methods are designed for classification tasks and do not align with our needs, and they also fail to capture the privacy risks associated with real training samples. The second line of research focuses on launching Membership Inference (MI) attacks on synthetic datasets, as seen in [7, 17, 18, 45], using the success rate of these attacks to estimate the likelihood of privacy breaches. However, the MIA proposed in [45] is designed to attack categorical attributes, and time-series datasets typically consist solely of continuous attributes. Thus, the MIA in [45] cannot attack synthetic time-series datasets. On the other hand, MIAs in [7, 17, 18] are designed for GANs with a single generator and discriminator. However, the structure of VFLGAN-TS is more complicated, with multiple generators and discriminators, and most discriminators are maintained by local participants, which are not accessible to potential attackers. Moreover, the shared discriminator alone cannot transform the raw data into an intermediate feature. Thus, these methods cannot be adapted to attack VFLGAN-TS directly. In summary, these existing MIAs are not applicable for evaluating the privacy risks of VFLGAN-TS. Previous research [60] introduces a more effective privacy auditing scheme, ASSD, which combines the shadow-model attack [41] and the leave-one-out setting [58], and significantly surpasses the performance of existing methods. However, ASSD is proposed for tabular data and does not work for time-series data, as it fails to find the most vulnerable training data. In this paper, we enhance ASSD for time-series data and use it for privacy analysis.

3 Preliminaries

This section provides preliminaries of GANs, the auditing scheme of VFLGAN [60], and differential privacy to facilitate a comprehensive understanding of the proposed VFLGAN-TS, DPVFLGAN-TS, and the enhanced auditing scheme.

3.1 GANs for Time-Series Data

This section begins with GANs for static data. Given a dataset $X \in \mathbb{R}^{N\times|A|}$ where N is the sample number and A is the attribute set, the generator of GAN, G, aims to generate synthetic data $\tilde{\boldsymbol{x}} = G(\boldsymbol{z}) \in \mathbb{R}^{|A|}$ and the distribution of the synthetic data $P_{\tilde{\boldsymbol{x}}}$ should be similar the distribution of the real data, $P_{\boldsymbol{x}}$ where $\boldsymbol{x} \in X$. The input $\boldsymbol{z}$ is a vector sampled from a simple distribution, such as the uniform or Gaussian distribution. The above objectives can be achieved with a discriminator, D. The generator and discriminator are trained through a competing game, where the discriminator is trained to distinguish $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. In contrast, the generator is trained to generate high-quality $\tilde{\boldsymbol{x}}$ to fool the discriminator. The game between the generator and the discriminator can be formally expressed as the following min-max objective [15],

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

where $D(\cdot)$ denotes the calculation of the discriminator D.

While static data only has the attribute dimension, time-series data, $X \in \mathbb{R}^{N\times|A|\times T}$, has both the attribute and temporal dimensions, where T denotes the time steps of each attribute. Normal GANs [14, 34, 37] with training objective (1) exhibit inadequate performance for generating time-series data [39, 59]. This inadequacy stems from their inability to differentiate effectively between correlations among temporal and feature dimensions. In [39], the authors proposed CosciGAN to solve this problem. In CosciGAN, there are $|A|$ attribute generators where A is the attribute set, $|A|$ attribute discriminators, and one central discriminator. Each attribute generator (e.g., $G_i, i \in \{1, \cdots, |A|\}$) is responsible for generating one synthetic attribute ($\tilde{\boldsymbol{x}}_i \in \mathbb{R}^{N\times T}$), and each attribute discriminator (e.g., $D_i, i \in \{1, \cdots, |A|\}$) is responsible for optimizing G_i to make the distribution of the synthetic attribute similar to that of real attribute, i.e., $P_{\tilde{\boldsymbol{x}}_i} \approx P_{\boldsymbol{x}_i}$. On the other hand, the central discriminator, D_C, is responsible for optimizing all generators to make the distribution of the synthetic data, $\tilde{\boldsymbol{x}} = [\tilde{\boldsymbol{x}}_1, \cdots, \tilde{\boldsymbol{x}}_{|A|}]$, similar to that of real data $\boldsymbol{x}$, i.e., $P_{\tilde{\boldsymbol{x}}} \approx P_{\boldsymbol{x}}$. Following the theoretical analysis in [15], the training objectives of CosciGAN can be expressed as,

$$\min_G \max_D \sum_{i=1}^{|A|} \left(\mathbb{E}[\log(D_i(\boldsymbol{x}_i))] + \mathbb{E}[\log(1 - D_i(\tilde{\boldsymbol{x}}_i))]\right) + \lambda \left(\mathbb{E}[\log(D_C(\boldsymbol{x}))] + \mathbb{E}[\log(1 - D_C(\tilde{\boldsymbol{x}}))]\right), \quad (2)$$

where $G = \{G_1, \cdots, G_{|A|}\}$, $D = \{D_1, \cdots, D_{|A|}, D_C\}$, and λ is a balancing coefficient.

3.2 Differential Privacy

Differential privacy (DP) [11] provides a rigorous privacy guarantee that can be quantitatively analyzed. The ϵ-DP is defined as follows.

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

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

The most commonly used DP in the literature is a relaxed version of the ϵ-DP, which allows the mechanism not to satisfy ϵ-DP with a small probability, δ. The relaxed version, (ϵ, δ)-DP [12], is defined as follows.

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

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

In [33], the α-Rényi divergences between $f(D)$ and $f(D')$ are applied to define Rényi Differential Privacy (RDP) which is a generalization of differential privacy. $(\alpha, \epsilon(\alpha))$-RDP is defined as follows.

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

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

The (R)DP budget should be accumulated if we apply multiple mechanisms to process the data sequentially as we train deep learning (DL) models for multiple iterations. We can calculate the accumulated RDP budget by the following proposition [33].

Proposition 1. *(Composition of RDP) Let $f : D \to R_1$ be (α, ϵ_1)-RDP and $g : R_1 \times D \to R_2$ be (α, ϵ_2)-RDP. Then the mechanism defined as (X, Y), where $X \sim f(D)$ and $Y \sim g(X, D)$, satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

According to the following proposition, RDP can be converted to (ϵ, δ)-DP and the proof can be found in [33].

Proposition 2. *(From RDP to (ϵ, δ)-DP) If f is an $(\alpha, \epsilon(\alpha))$-RDP mechanism, it also satisfies $\left(\epsilon(\alpha) + \frac{log 1/\delta}{\alpha - 1}, \delta\right)$-DP for any $0 < \delta < 1$.*

According to Proposition 2, given a δ we can get a tight (ϵ', δ)-DP bound by

$$\epsilon' = \min_\alpha \left(\epsilon(\alpha) + \frac{log 1/\delta}{\alpha - 1}\right). \quad (3)$$

A tighter upper bound on RDP is provided in [52] by considering the combination of the subsampling procedure and random mechanism. This is important for differentially private DL since DL models are mostly updated according to a subsampled mini-batch of data. The enhanced RDP bound can be calculated according to the following proposition, and the proof can be found in [52].

Proposition 3. *(RDP for Subsampled Mechanisms). Given a dataset of n points drawn from a domain $\mathcal{X}$ and a (randomized) mechanism $\mathcal{M}$ that takes an input from $\mathcal{X}^m$ for $m \leq n$, let the randomized algorithm $\mathcal{M}\circ$ subsample be defined as (1) subsample: subsample without replacement m datapoints of the dataset (sampling rate $\gamma = m/n$), and (2) apply $\mathcal{M}$: a randomized algorithm taking the subsampled dataset as the input. For all integers $\alpha \geq 2$, if $\mathcal{M}$ obeys $(\alpha, \epsilon(\alpha))$-RDP, then this new randomized algorithm $\mathcal{M}\circ$ subsample obeys $(\alpha, \epsilon'(\alpha))$-RDP where,*

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

4 Proposed VFLGAN-TS

This section begins with the problem formulation of the publication of vertically partitioned time-series data. Next, we introduce the framework of VFLGAN-TS and separate the problem into two distinct objectives to learn the correlation along temporal and attribute dimensions. Then, we describe the training process of VFLGAN-TS and DPVFLGAN-TS. Last, we introduce the enhanced auditing scheme.

4.1 Problem Formulation

In this paper, we consider a scenario including M non-colluding parties. Each party, P_i, maintains a private time-series dataset, $X_i \in \mathbb{R}^{N\times|A_i|\times T}$ where N denotes the sample number, A_i denotes the attribute set of X_i, and T denotes the time-series length. This is a novel research problem, i.e., the publication of vertically partitioned time-series data, which is different from previous papers [22, 60] that focus on generating tabular datasets, $X_i \in \mathbb{R}^{N\times|A_i|}$. On the other hand, [14, 34, 37] can only work in centralized scenarios for time-series data generation. This paper has the following two assumptions for this scenario.

Assumption 1. *Samples in each private dataset with the same index correspond to the same object. This alignment can be achieved by using private set intersection protocols [8, 19].*

Assumption 2. *There is no common attribute in different parties.*

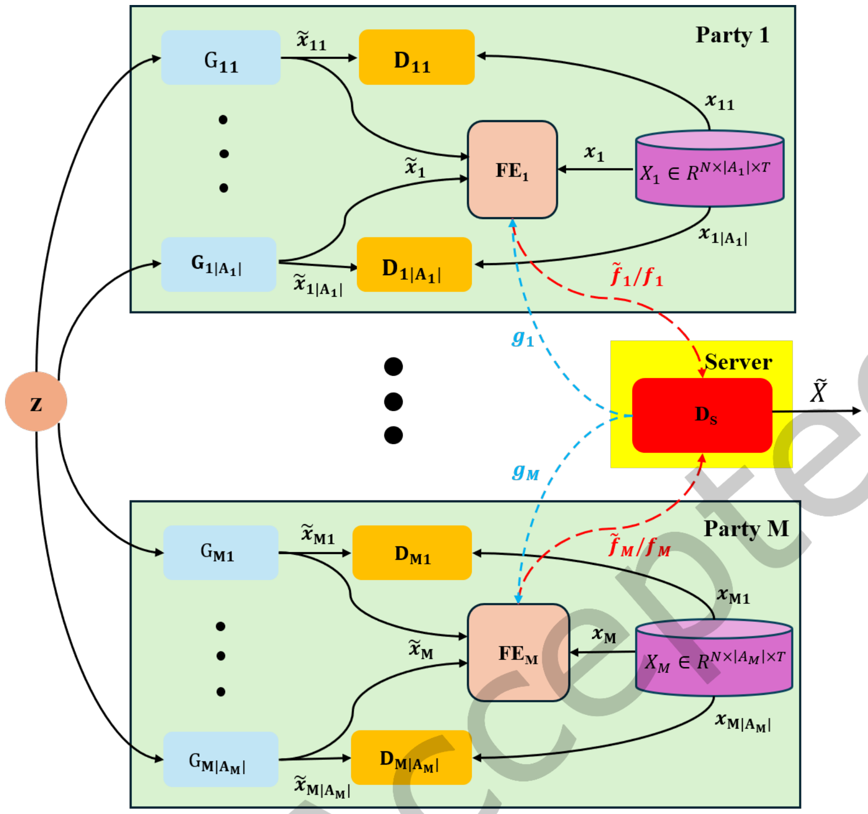
We can obtain a new dataset with the above assumptions by combining the M private datasets, i.e., $X = [X_1, \cdots, X_M]$. The objectives of vertically partitioned data publication are to release a synthetic dataset $\tilde{X}$ without sharing the local private data and the sample ($\tilde{\boldsymbol{x}} \in \tilde{X}$) in the synthetic dataset should follow a similar distribution to the sample ($\boldsymbol{x} \in X$) in the private dataset. The problem can be formulated as,

$$\min_{\theta} \mathcal{D}(P_{\tilde{\boldsymbol{x}}}, P_{\boldsymbol{x}}), \tag{4}$$

where θ denotes the parameters to generate $\tilde{\boldsymbol{x}}$ and $\mathcal{D}$ is any suitable measure of the distance between two distributions.

4.2 Framework of VFLGAN-TS

The framework of VFLGAN-TS is depicted in Fig. 1. VFLGAN-TS integrates the attribute discriminator of CosciGAN [39] and VFL [27] to achieve the objectives (4). Note that VFL is a concept rather than a specific method. In this paper, we propose the first vertical federated learning framework for the publication of vertically partitioned time-series data, i.e., VFLGAN-TS. As shown in Fig. 1, each party in the framework maintains a local private dataset (X_i) with attribute set A_i, $|A_i|$ attribute generators, and $|A_i|$ attribute discriminators. In the local party P_i, each attribute generator, G_{ij}, is to generate a synthetic attribute ($\tilde{\boldsymbol{x}}_{ij}$), and the attribute discriminator,



X_i: Private dataset of Party; $\boldsymbol{x_i}$: Sample of X_i; $\tilde{\boldsymbol{x}}_i$: Synthetic sample of X_i;
$\boldsymbol{x_{ij}}$: Attribute j of sample $\boldsymbol{x_i}$; $\tilde{\boldsymbol{x}}_{ij}$: Attribute j of sample $\tilde{\boldsymbol{x}}_i$; $\tilde{X}$: Synthetic dataset;
G_{ij}: Attribute generator for $\tilde{\boldsymbol{x}}_{ij}$; D_{ij}: Attribute discriminator for G_{ij}; D_S: Shared discriminator;
FE_i: Feature extractor of Party i; $\tilde{\boldsymbol{f}}_i/\boldsymbol{f}_i$: feature of $\tilde{\boldsymbol{x}}_i/\boldsymbol{x}_i$; $\boldsymbol{g}_i$: Gradients for FE_i; **z**: Aligned noise vector.

Fig. 1. Framework of the Proposed VFLGAN-TS.

D_{ij}, is to optimize G_{ij} to make the distribution of each synthetic attribute ($\tilde{\boldsymbol{x}}_{ij}$) to be similar to the distribution of the corresponding real attribute ($\boldsymbol{x}_{ij}$), i.e.,

$$\min_{\theta_{G_{ij}}} \mathcal{D}(P_{G_{ij}(\boldsymbol{z})}, P_{\boldsymbol{x}_{ij}}), \quad \text{where } \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{5}$$

Notably, the random vector $\boldsymbol{z}$ is the same for all attribute generators, which can be achieved by a pseudorandom number generator. According to (5), the local discriminators are to learn the temporal correlation within one attribute. On the other hand, the shared discriminator in the server is to learn the correlation among all attributes. Unlike CosciGAN, we utilise a trainable feature extractor (FE) in each party to extract features from local attributes and send these features to the server, thereby avoiding the transmission of private data to the server and protecting privacy. Then, the features from all parties are concatenated in the server and discriminated by the shared discriminator (D_S). D_S is to optimize all attribute generators simultaneously to make the distribution of synthetic data ($\tilde{\boldsymbol{x}}$) to be similar to that of real data ($\boldsymbol{x}$), i.e.,

$$\min_{\theta_G} \mathcal{D}(P_{\tilde{\boldsymbol{x}}}, P_{\boldsymbol{x}}), \quad \text{where } \tilde{\boldsymbol{x}} = [G_{11}\ (\boldsymbol{z}) \cdots, G_{M|A_M|}(\boldsymbol{z})],\ \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),\ \text{and } G = \{G_{11} \cdots, G_{M|A_M|}\}. \tag{6}$$

Takeaway: In the proposed VFLGAN-TS, the objectives of the vertically partitioned time-series data publication (4) are transformed to simultaneously optimize (5) and (6). Although the objectives of the shared discriminator can learn the correlation along both temporal and attribute dimensions, it primarily focuses on the attribute dimension, as the local discriminators learn the correlation along the temporal dimension for each attribute. The separate treatment of the correlation along temporal and attribute dimensions is the key reason VFLGAN-TS fits the time-series data significantly better than VFLGAN [60]. In a real-world scenario, a bank and an online shopping platform with common users can collaborate to publish a synthetic time-series dataset that combines attributes from both parties. By analysing the synthetic dataset, they can capture users' financial and consumption dynamics, supporting applications such as personalised recommendation, joint risk assessment, and credit scoring while preserving data privacy since no raw data is shared. In this two-party setting, the bank can be denoted as Party 1 and the online shopping platform is denoted as Party 2 in Fig. 1. The bank can contribute financial behaviour time-series data, such as account balance histories, transaction records, credit card and loan repayment histories, and investment portfolio changes, which are denoted as $x_{11}-x_{14}$ in Fig. 1. Meanwhile, online shopping platforms can provide consumer behaviour time-series data such as browsing histories, shopping cart activities, purchase and return records, and review activities, which are denoted as $x_{21}-x_{24}$ in Fig. 1. During inference, the synthetic dataset combined at the Server in Fig. 1 contains all attributes, i.e., $x_{11}-x_{14}$ and $x_{21}-x_{24}$. As analyzed in Section 4.1, this scenario cannot be solved properly by the methods proposed in [22, 60] since they are proposed for tabular datasets, nor by the methods proposed in [14, 34, 37] since they cannot handle the vertically distributed data. This paper is the first to study the publication of vertically partitioned time-series data.

4.3 Training Process of VFLGAN-TS

The generators in the VFLGAN-TS, which are deep neural networks, require gradient updates to achieve the objectives of VFLGAN-TS in (5) and (6). However, since the above objectives are non-differential, they cannot be directly utilized to train the model. According to (1), we can convert the objectives (5) into the following min-max optimization objectives to update the parameters of the attribute generator G_{ij} and attribute discriminator D_{ij},

$$\min_{G_{ij}} \max_{D_{ij}} \mathbb{E}[\log(D_{ij}(\boldsymbol{x}_{ij}))] + \mathbb{E}[\log(1 - D_{ij}(G_{ij}(\boldsymbol{z})))], \quad \text{where } \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{7}$$

where the subscript ij means the model is responsible for Attribute j of Party i and $\boldsymbol{x}_{ij}$ denotes the real Attribute j of Party i. Similarly, to achieve objective (6), all real and synthetic attributes should be considered. However, local attributes cannot be shared across parties due to the privacy constraints of the vertically partitioned scenario. To avoid sharing the raw data, we apply a local feature extractor in each party to process real and synthetic data and transmit the feature to the shared discriminator in the server, as shown in Fig. 1. The min-max optimization objectives between the attribute generators in all parties (G) and the shared discriminator (D_S) in the server can be described as,

$$\begin{aligned} &\min_{G} \max_{D_S} \mathbb{E}[\log(D_S([\boldsymbol{f}_1, \cdots, \boldsymbol{f}_M]))] + \mathbb{E}[\log(1 - D_S([\tilde{\boldsymbol{f}}_1, \cdots, \tilde{\boldsymbol{f}}_M]))], \\ &\text{where } \boldsymbol{f}_i = FE_i(\boldsymbol{x}_i),\ \tilde{\boldsymbol{f}}_i = FE_i(\tilde{\boldsymbol{x}}_i),\ \text{and } i \in \{1, \cdots, M\}. \end{aligned} \tag{8}$$

In (8), FE_i is the feature extractor in Party i, $\boldsymbol{x}_i$ is the real data of Party i, and $\tilde{\boldsymbol{x}}_i$ denotes the synthetic data of Party i.

According to (7), the loss function for the local attribute discriminator (D_{ij}) can be described as,

$$\mathcal{L}_{D_{ij}} = -(\mathbb{E}[\log(D_{ij}(\boldsymbol{x}_{ij}))] + \mathbb{E}[\log(1 - D_{ij}(G_{ij}(\boldsymbol{z})))]). \tag{9}$$

The loss function of D_{ij} is the negative of objective (7) since the objective of D_{ij} is to maximize (7) while gradient descent methods in deep learning framework are used to optimize the model by minimizing the loss function. Similarly, according to (8), the loss function of D_S can be described as,

$$\begin{aligned}&\mathcal{L}_{D_S} = -(\mathbb{E}[\log(D_S([\boldsymbol{f}_1, \cdots, \boldsymbol{f}_M]))] + \mathbb{E}[\log(1 - D_S([\tilde{\boldsymbol{f}}_1, \cdots, \tilde{\boldsymbol{f}}_M]))]),\\ &\text{where } \boldsymbol{f}_i = EF_i(\boldsymbol{x}_i),\ \tilde{\boldsymbol{f}}_i = EF_i(\tilde{\boldsymbol{x}}_i),\ \text{and } i \in \{1, \cdots, M\}.\end{aligned} \tag{10}$$

Note that the feature extractors (EF_i) are also optimized according to (10) since they are parts of D_S.

On the other hand, since the local attribute generator (e.g., G_{ij}) is required to learn the correlation along both temporal and attribute dimensions, the loss function of G_{ij}, $\mathcal{L}_{G_{ij}}$, should consider the objectives of local attribute discriminator and the shared discriminator and thus can be described as,

$$\mathcal{L}_{G_{ij}} = \mathbb{E}[\log(1 - D_{ij}(G_{ij}(\boldsymbol{z})))] + \beta\mathbb{E}[\log(1 - D_S([\tilde{\boldsymbol{f}}_1, \cdots, \tilde{\boldsymbol{f}}_M]))]), \tag{11}$$

where G_{ij} contributes to the $\tilde{\boldsymbol{f}}_i$ in the second term and β is a balancing coefficient.

According to the loss functions (9), (10), and (11), the gradients of parameters of the shared discriminator (D_S), local attribute discriminator (D_{ij}), local attribute generator (G_{ij}), and local feature extractor (FE_i), can be calculated by,

$$\mathcal{G}_{D_S} = \nabla_{\boldsymbol{\theta}_{D_S}} \mathcal{L}_{D_S}, \tag{12}$$

$$\mathcal{G}_{D_{ij}} = \nabla_{\boldsymbol{\theta}_{D_{ij}}} \mathcal{L}_{D_{ij}}, \tag{13}$$

$$\mathcal{G}_{G_{ij}} = \nabla_{\boldsymbol{\theta}_{G_{ij}}} \mathcal{L}_{G_{ij}}, \tag{14}$$

$$\mathcal{G}_{FE_i} = \nabla_{\boldsymbol{\theta}_{FE_i}} \mathcal{L}_{D_S}, \tag{15}$$

$$\text{where} \quad i \in \{1, \cdots, M\}, j \in \{1, \cdots, |A_i|\}.$$

After getting the gradients, we optimise the parameters by applying Adam [24] optimizer.

The training process of VFLGAN-TS is summarized in Algorithm 1. In each training iteration, we first train the discriminators and feature extractors. We subsample a mini-batch of the aligned attributes in each party and generate a mini-batch of synthetic attributes with local attribute generators. The local attribute discriminators are trained to distinguish the real and synthetic attributes. The shared discriminator and feature extractors are trained to distinguish the combination of local real and synthetic attributes. Then, we train the attribute generators to generate more realistic synthetic attributes. The above steps repeat for T_{max} iterations, and the algorithm outputs trained local attribute generators.

4.4 Mechanism of VFLGAN-TS

In Theorem 4.1, we demonstrate that the proposed training process can lead to the same optimal solution as objective (4). Furthermore, in Theorem 4.2, we establish the convergence of the proposed training process.

Theorem 4.1. *(Optimality of VFLGAN-TS) The global optimal point of VFLGAN-TS training objective is achieved if and only if $P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}$, namely, the distribution of real and synthetic data are identical.*

Proof. The proof is presented in Appendix A. □

Theorem 4.2. *(Convergence of VFLGAN-TS) If all VFLGAN-TS networks ($G_{ij}, D_{ij}, FE_i, D_S, \forall i \in \{1, 2, \ldots, M\}, j \in \{1, 2, \ldots, |A_i|\}$) have enough capacity, and at each step the discriminators (i.e., D_{ij}, FE_i, D_S) are allowed to reach their optimum given the (corresponding) generators (G_{ij}, G), and $P_{\tilde{\mathbf{x}}} = P_{G(\boldsymbol{z})}$ is updated to improve the criterion in (38) (LHS), then $P_{\tilde{\mathbf{x}}} = P_{G(\boldsymbol{z})}$ converges to $P_{\mathbf{x}}$.*

Proof. The proof is presented in Appendix A. □

Algorithm 1: Training Process of (DP-)VFLGAN-TS

```
Input  : $G_{ij}$: attribute generator for Attribute $j$ of Party $i$; $D_{ij}$: attribute discriminator for $G_{ij}$; $D_S$: shared discriminator; $FE_i$:
         feature extractor in Party $i$; $X_i$: dataset in Party $i$; $A_i$: attribute set of $X_i$; $M$: number of parties; $\mathcal{G}$: parameters gradients; $\mathcal{G}^1$:
         gradients of the first-layer parameters; $B$: batch size; $l$: latent dimension.
Output : Trained $\{G_{11}, \cdots, G_{M|A_M|}\}$.
1  Initialize all generators and discriminators;
2  for $epoch$ in $\{1, 2, \cdots, T_{max}\}$ do
       // update discriminators (line 3 to 22);
3      $x^B = [x_1^B, \cdots, x_M^B]$ where $x_i^B \subset X_i$ // Subsample a mini-batch of aligned data in each party;
4      Generate $z^B \sim \mathcal{N}(0,1)^{l \times B}$
5      for $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$ do
6          $\tilde{x}_{ij}^B = G_{ij}(z^B)$ // Generate synthetic attribute;
7      for $i \in \{1, \cdots, M\}$ do
8          $\tilde{f}_i^B = FE_i([\tilde{x}_{i1}, \cdots, \tilde{x}_{i|A_i|}])$
9          $f_i^B = FE_i([x_{i1}, \cdots, x_{i|A_i|}])$ // Extract features;
10     for $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$ do
11         $\mathcal{L}_{D_{ij}} = -(\mathbb{E}[\log(D_{ij}(x_{ij}))] + \mathbb{E}[\log(1 - D_{ij}(G_{ij}(z)))])$ // Compute loss function of $D_{ij}$ [39];
12     $\mathcal{L}_{D_S} = -(\mathbb{E}[\log(D_S([f_1, \cdots, f_M]))] + \mathbb{E}[\log(1 - D_S([\tilde{f}_1, \cdots, \tilde{f}_M]))])$// Compute loss function of $D_S$;
13     for $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$ do
14         $\mathcal{G}_{D_{ij}} = \nabla_{\theta_{D_{ij}}} \mathcal{L}_{D_{ij}}$ // Compute gradients of $D_{ij}$;
15         if Training a differentially private version then
16             $\mathcal{G}_{D_{ij}}^1 = clip(\mathcal{G}_{D_{ij}}^1, C) + \mathcal{N}\left(0, \sigma^2(2C)^2 I\right)$
17     $\mathcal{G}_{D_S} = \nabla_{\theta_{D_S}} \mathcal{L}_{D_S}$ // Compute gradients of $D_S$;
18     for $i \in \{1, \cdots, M\}$ do
19         $\mathcal{G}_{FE_i} = \nabla_{\theta_{FE_i}} \mathcal{L}_{D_S}$ // Compute gradients of $FE_i$;
20         if Training a differentially private version then
21             $\mathcal{G}_{FE_i}^1 = clip(\mathcal{G}_{FE_i}^1, C) + \mathcal{N}\left(0, \sigma^2(2C)^2 I\right)$
22     Apply Adam optimizer with $\mathcal{G}_{D_{ij}}$, $\mathcal{G}_{D_S}$, and $\mathcal{G}_{FE_i}$ to update the parameters of $D_{ij}$, $D_S$, and $FE_i$, where $i \in \{1, \cdots, M\}$ and
       $j \in \{1, \cdots, |A_i|\}$.
       // update generators (line 25 to 35);
23     Generate $z^B \sim \mathcal{N}(0,1)^{l \times B}$
24     for $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$ do
25         $\tilde{x}_{ij}^B = G_{ij}(z^B)$ // Generate synthetic attribute;
26     for $i \in \{1, \cdots, M\}$ do
27         $\tilde{f}_i^B = FE_i([\tilde{x}_{i1}, \cdots, \tilde{x}_{i|A_i|}])$
28         $f_i^B = FE_i([x_{i1}, \cdots, x_{i|A_i|}])$ // Extract features;
29     for $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$ do
30         $\mathcal{L}_{G_{ij}} = \mathbb{E}[\log(1 - D_{ij}(G_{ij}(z)))] + \beta\mathbb{E}[\log(1 - D_S([\tilde{f}_1, \cdots, \tilde{f}_M]))])$// Loss function of $G_{ij}$;
31     for $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$ do
32         $\mathcal{G}_{G_{ij}} = \nabla_{\theta_{G_{ij}}} \mathcal{L}_{G_{ij}}$ // Compute gradients of $G_{ij}$;
33     Apply Adam optimizer with $\mathcal{G}_{G_{ij}}$ to update the parameters of $G_{ij}$, where $i \in \{1, \cdots, M\}$ and $j \in \{1, \cdots, |A_i|\}$.
34 Return $\{G_{11}, \cdots, G_{M|A_M|}\}$.
```

Assume M parties in the VFL framework and each party i maintain a local dataset $X_i \in \mathbb{R}^{N\times|A_i|\times T}$. Then, each attribute of the local dataset X_i is a time series of length T, i.e., $X_a = \{x_{a1}, x_{a2}, \cdots, x_{aT}\}$ where $a \in \{1, 2, \cdots, |A_i|\}$. According to Theorem 4.1 and Theorem 4.2, after VFLGAN-TS converges, the generated synthetic attribute $\tilde{X}_a$ follows a similar joint distribution of the real attribute X_a among T time steps, with the help of the corresponding local attribute discriminator. Besides, the concatenation of all synthetic local attributes, i.e., the generated synthetic dataset $\tilde{X} = [\tilde{X}_1, \tilde{X}_2, \cdots, \tilde{X}_M]$ follows a similar joint distribution of the real dataset along both time-step and attribute dimensions, with the help of the shared discriminator in the server. While the shared server is capable of modelling the joint distribution over time-step and attribute dimensions, jointly learning both dimensions has been shown to impede convergence [39]. In VFLGAN-TS, since the joint distribution over the time-step dimension of each attribute is learned by the attribute discriminator, the shared discriminator primarily learns the joint distribution over the attribute dimension. As a result, the correlation between different time steps within each attribute is modelled by the corresponding attribute discriminator, and the correlation across different attributes is modelled by the shared discriminator.

4.5 Differentially Private VFLGAN-TS

The training process of differentially private VFLGAN-TS (DPVFLGAN-TS) is also summarised in Algorithm 1 since the training processes of both versions are the same for most steps. There are two main differences between the two training processes. (i) We adapt the DPSGD [1] to train DPVFLGAN-TS. When updating the parameters of the first linear layers of the local attribute discriminators and feature extractors, we clip and add Gaussian noise to the gradients as follows,

$$clip(\mathcal{G}^1_{model}, C) = \mathcal{G}^1_{model}/\max\left(1, \left\|\mathcal{G}^1_{model}\right\|_2/C\right), \tag{16}$$

$$\mathcal{G}^1_{model} = clip(\mathcal{G}^1_{model}, C) + \mathcal{N}\left(0, \sigma^2(2C)^2 I\right), \tag{17}$$

where the subscript *model* denotes any local attribute discriminator and feature extractor, $\mathcal{G}^1_{model}$ denotes the gradients of the first layer parameters of the *model*, and C denotes the clipping bound. In Theorem 4.3, we prove that clipping and adding Gaussian noise to the gradients of the first-layer parameters can provide a DP guarantee for the local attribute discriminators, attribute generators, and feature extractors. Note that the DP budget in our framework is different from the previous variants of DPSGD [6, 22, 60] due to the novel distributed system structure. (ii) We need to set a privacy budget, i.e., (ϵ, δ)-DP, before training DPVFLGAN-TS. Here, we apply the official implementation of [52] to select the proper training iterations (T_{max} in Algorithm 1) and σ in (17) to achieve the privacy budget, i.e., (ϵ, δ)-DP.

THEOREM 4.3. *(RDP Guarantee) All the local attribute discriminators and feature extractors satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP and the local attribute generators satisfy $(\alpha, \alpha/\sigma^2)$-RDP in one training iteration of DPVFLGAN-TS.*

PROOF. The proof is presented in Appendix B. □

Now, we introduce a method to select appropriate σ and T_{max} to meet the DP budget using Theorem 4.3. First, the RDP guarantee in Theorem 4.3 can be enhanced by Proposition 3 for external attackers and the server since we subsample mini-batch records from the whole training dataset in Algorithm 1. Then, the RDP budget is accumulated by T_{max} iterations, which can be calculated according to Proposition 1. Last, the RDP guarantee is converted to DP guarantee using (3). We can adjust the σ and T_{max} to ensure the calculated DP guarantee aligns with our DP budget. Notably, similar to the DP guarantee in [22, 60], the above DP guarantee specifically addresses external threats. This is due to the deterministic nature of mini-batch selection in each training iteration for internal parties (excluding the server), as opposed to a subsampling process. For internal adversaries, privacy can be enhanced by sacrificing efficiency. For example, the mini-batch size can be changed to $\hat{B} > B$. When updating the parameters, each party can randomly select the gradients of B samples and mask the gradients of other $(\hat{B} - B)$ samples. In this way, all parties do not know precisely which samples others use to update the parameters.

4.6 Privacy Auditing for VFLGAN-TS

In this section, we enhance the ASSD proposed in [60] since it is not effective for the synthetic datasets generated by VFLGAN-TS, as shown in Table 7. The key objective of MIA is to identify vulnerable samples in the training dataset. However, ASSD was initially designed for tabular data and performs poorly in identifying vulnerable samples when applied to time-series datasets. To address this limitation, we enhance ASSD in two aspects: target sample selection and feature extraction. By integrating these enhancements into ASSD, we develop the first MIA specifically tailored for time-series synthetic datasets. We apply the same threat model as [60]; details are available in **Appendix C**.

We propose two methods to select the target sample. The first method is to choose the target sample (x_t) with the maximum nearest neighbour distance, i.e.,

$$\arg\max_{\boldsymbol{x}\in X} \min_{\boldsymbol{x'}\in X\setminus\boldsymbol{x}} ||\boldsymbol{x} - \boldsymbol{x'}||_n, \tag{18}$$

where $||\cdot||_n$ denotes the n-norm function. In the second method, we first construct a candidate set S_c by selecting m samples with the first m maximum nearest neighbour distances,

$$S_c = \text{Top-}m\{\min_{\boldsymbol{x'}\in X\setminus\boldsymbol{x}} ||\boldsymbol{x} - \boldsymbol{x'}||_n\}. \tag{19}$$

Then, we train a VFLGAN-TS with the whole training dataset and use it to generate a synthetic dataset $\tilde{X}$. Then, the target sample (x_t) is selected by,

$$\arg\min_{\boldsymbol{x}\in S_c} \text{sum}(\text{minimum-}k\{\min_{\boldsymbol{x'}\in\tilde{X}} ||\boldsymbol{x} - \boldsymbol{x'}||_n\}), \tag{20}$$

where $\text{sum}(S)$ denotes sum of all elements in set S.

Takeaway: (18) is to select the most out-of-distribution sample while (20) is to select the most influential training sample for the synthetic dataset. Some papers [31, 41] assume that the most out-of-distribution samples are the most vulnerable to privacy attacks. However, we found that in some cases, the most out-of-distribution sample has only a minor influence on the generative model and is thus difficult to attack.

According to (20), we propose the KNN feature, the sum of the k nearest neighbour distances between the target sample (x_t) and samples of the synthetic dataset $(\tilde{X})$,

$$F_{\text{KNN}} = \text{sum}(\text{minimum-}k\{\min_{\boldsymbol{x'}\in\tilde{X}} ||\boldsymbol{x_t} - \boldsymbol{x'}||_n\}). \tag{21}$$

For the KNN feature, we compute the AUC-ROC to estimate the privacy breaches using synthetic datasets, rather than training a random forest as described in [60]. This is because the KNN feature is a single value and AUC-ROC is better for representing the difference between $F_{0_{1:M}}$ and $F_{1_{1:M}}$ than splitting the features into training and test sets and training a random forest on the training set and evaluating the accuracy of the random forest on the test set.

5 Experiments and Results

This section first introduces the experimental environment, then presents the evaluation results of VFLGAN-TS, and last, gives a privacy analysis of the proposed framework and generated synthetic datasets.

5.1 Experimental Environments

5.1.1 Datasets. We conduct experiments with two synthetic datasets and three real-world datasets.

Synthetic Sine Datasets: Similar to [21, 39], the two synthetic datasets are constructed with sine signals. The first synthetic dataset is named two-attribute Sine dataset, which is constructed as follows,

$$F_{11} = Asin(2\pi f_1 \boldsymbol{t}) + \epsilon, \quad F_{21} = Asin(2\pi f_2 \boldsymbol{t}) + \epsilon, \tag{22}$$

where F_{ij} denotes the attribute j stored in Party i, A can be sampled from $\mathcal{N}(0.4, 0.05)$ (class 1) or $\mathcal{N}(0.6, 0.05)$ (class 2) with equal propability, $\epsilon \in \mathcal{N}(0, 0.05)$, and f_1 and f_2 are set 0.01 and 0.005, respectively. In summary, in the two-party scenario, each party store one attribute, and the amplitudes of two attributes are the same for a given sample. We can utilize this characteristic, i.e., the same amplitude, to evaluate whether the generative methods learn the correlation between the two attributes stored in different parties. We generate 1,024 samples for both class, i.e., $A \in \mathcal{N}(0.4, 0.05)$ and $A \in \mathcal{N}(0.6, 0.05)$, and each attribute consists of 800 time steps ($\boldsymbol{t} = [0, 1, \cdots, 799]$ in (22)).

The second synthetic dataset, named six-attribute Sine dataset, also contains two classes of samples and six attributes to evaluate the scalability of our methods. The construction details can be found in **Appendix D**. In the two-party scenario, each party holds three attributes. Similar to (22), the frequency of each attribute is different and the amplitudes of the six attributes are the same for a given sample.

EEG Eye State Dataset: EEG dataset [38] contains 14 attributes and a label indicating whether the patient's eyes were open or closed. Similar to [39], we select five attributes that achieve the best performance on the classification task and split the datasets into two datasets according to the label, i.e., EEG 0 and EEG 1. In the two-party scenario, one party has three attributes, and the other has two.

Stock Dataset: Stock dataset consists of Google stock price data from 2004 to 2019 and includes six attributes. We apply the same preprocessing method as [59] and obtain a dataset containing 3661 frames, and each frame contains six attributes with 24 time steps. In the two-party scenario, each party has three attributes.

Energy Dataset: Energy dataset [5] contains 19712 frames, and each frame consists of 28 attributes with 24 time steps. In the two-party scenario, each party has 14 attributes.

5.1.2 Evaluation Metrics. In this paper, we apply Wasserstein distance, performance on downstream tasks, and visualization for evaluation.

Wasserstein distance: The Wasserstein distance measures the similarity between two probability distributions [53]. A lower Wasserstein distance means there is more similarity between the two distributions. We calculate the Wasserstein distance between the attributes of the synthetic and real datasets to evaluate the performance of different methods. Assuming there are N attributes and each attribute is a time series with T time steps, we calculate the Average Wasserstein Distance (AWD) as,

$$\text{AWD} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{n=1}^{N} \text{WD}(F_n^t, \tilde{F}_n^t), \tag{23}$$

where F_n^t and $\tilde{F}_n^t$ represent the combination of attribute n at time step t across all samples in the real and synthetic datasets, respectively, and WD[1] is a function to estimate the Wasserstein distance. Although AWD cannot measure the correlation among different attributes or time steps, it has two benefits. First, its rapid computation allows the evaluation after each training epoch, and we can thus select the model with the minimum AWD as the final model. Second, it is privacy-preserving since it requires no information exchange between different parties.

[1]We use SciPy library to compute the Wasserstein distance and details can be found in https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wasserstein_distance.html#scipy.stats.wasserstein_distance

Performance on Downstream Tasks: Performance on Downstream Tasks (PDT) is to assess the prospects of synthetic datasets in real-world applications. Different from [21, 39] and [59] that only consider the training on synthetic data and testing on real data scenarios, we consider the following four scenarios like [60]: (i) training on real data and testing on real data (TRTR), (ii) training on synthetic data and testing on synthetic data (TSTS), (iii) training on real data and testing on synthetic data (TRTS), (iv) and vice versa (TSTR). The TRTR setting serves as a baseline for PDT. The performance in TSTS, TRTS, and TSTR scenarios should be similar to that in the TRTR scenario if the distribution of synthetic data is similar to that of real data. Thus, we use Total Performance Difference (TPD) as the final comparison metric,

$$\text{TPD} = \sum_{i \in \{TSTS, TRTS, TSTR\}} (P_i - P_{TRTR}), \tag{24}$$

where P denotes performance.

Visualization: The visualization results can intuitively show how closely the distribution of generated samples resembles that of the real samples in two-dimensional space. For visualization, we project synthetic and real datasets into a two-dimensional space through t-SNE [46] and PCA [4] analysis.

Autocorrelation function (ACF): ACF is a fundamental concept in time series analysis that measures the correlation between observations of a time series separated by different time lags, which is calculated by,

$$\rho(k) = \frac{\text{Cov}(X_t, X_{t-k})}{\sigma^2}, \tag{25}$$

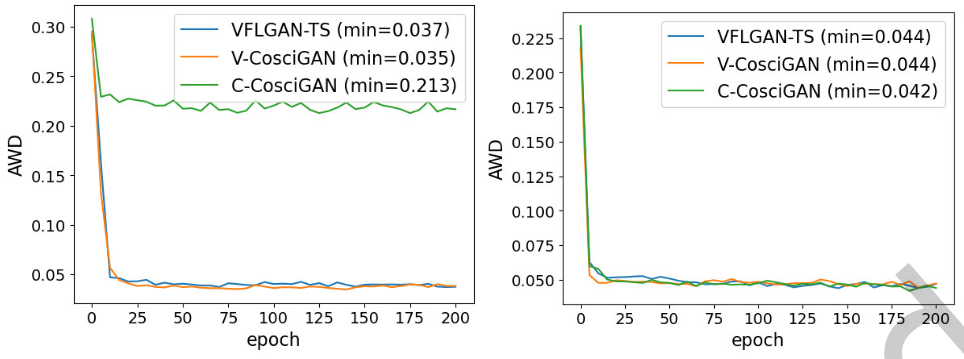
where σ^2 is the variance of the time series X. Similar CFs between two time series suggest that they share similar temporal dependence structures. Considering all the attributes and time lags, we calculate the Average Absolute Difference of ACFs (AADA) between real and synthetic time series by,

$$\text{AADA} = \sum_{a=1}^{|A|} \sum_{k=1}^{T/2} \left| \frac{\text{Cov}(X_{at}, X_{a(t-k)})}{\sigma^2} - \frac{\text{Cov}(\tilde{X}_{at}, \tilde{X}_{a(t-k)})}{\tilde{\sigma}^2} \right|, \tag{26}$$

where T is the length of the time series and A is the attribute set of all participants. A smaller AADA suggests a greater similarity between the real and synthetic time series.

5.1.3 Baselines. We compare VFLGAN-TS with VFLGAN [60] to show the improvement. We also consider two CosciGAN-based baselines to evaluate the proposed VFLGAN-TS since VFLGAN-TS is constructed based on CosciGAN [39]. The first baseline is the CosciGAN trained in a centralized manner, i.e., the model can access all attributes of the dataset, which is named Centralized CosciGAN (C-CosciGAN). The other baseline is the CosciGAN trained in a vertically partitioned manner, i.e., the Vertical CosciGAN (V-CosciGAN). In V-CosciGAN, each party trains a local CosciGAN that can only access the local attributes. In the inference process, the synthetic dataset is constructed with the synthetic attributes generated by each local CosciGAN.

Takeaway: C-CosciGAN can represent the upper limit of VFLGAN-TS since it is trained centrally. By comparing VFLGAN-TS with C-CosciGAN, we can evaluate the performance degradation associated with the vertically partitioned scenario. On the other hand, C-CosciGAN can represent the lower limit of VFLGAN-TS since the generators are trained locally. Comparing VFLGAN-TS to V-CosciGAN allows us to assess the performance gains attributed to the VFL framework over a straightforward adaptation of CosciGAN to the vertically partitioned scenario. Additionally, for a fair comparison, the structure of the attribute generators and attribute discriminators is identical for VFLGAN-TS, VFLGAN, C-CosciGAN, and V-CosciGAN. Besides, here we aim to align the structure and computation of the shared discriminators in VFLGAN and VFLGAN-TS with those of the central discriminators in C-CosciGAN and V-CosciGAN.



(a) Two-attribute Sine (b) Six-attribute Sine

Fig. 2. Wasserstein distance curves during training different methods on Sine Datasets.

5.2 Evaluation Results

This section shows the evaluation results based on the above-mentioned experimental environment.

5.2.1 Sine Datasets. Given a sample from Sine Datasets, the amplitude of each feature is the same. Therefore, we utilise this property to assess the quality of Sine datasets. Different from (23), to calculate Wasserstein Distance (WD), we first estimate the amplitude vector ($\boldsymbol{a}$) of each sample in real and synthetic datasets by,

$$\boldsymbol{a} \triangleq [a_1, \cdots, a_N] = [\mathcal{A}(f_1), \cdots, \mathcal{A}(f_N)], \tag{27}$$

where a_1 denotes the estimated amplitude of feature f_1 and $\mathcal{A}$ denotes the estimation function. Then, we calculate the AWD of the amplitude distributions of each feature between real and synthetic datasets by,

$$\text{AWD} = \sum_{n=1}^{N} \text{WD}(A_n, \tilde{A}_n), \tag{28}$$

where A_n is the vector that combines the amplitudes a_i of all samples in the real dataset and $\tilde{A}_n$ is the vector that combines the amplitudes $\tilde{a}_n$ of all samples in the synthetic dataset. We do not need to calculate the average Wasserstein distance across the time dimension like (23) since the time-series features have been converted to amplitude features. Figure 2 shows the AWD curves of the three methods during the training process on two-feature and six-feature Sine datasets. As shown in Fig. 2, the C-CosciGAN performs poorly on the two-feature Sine dataset while the proposed VFLGAN-TS performs similarly to V-CosciGAN. Besides, VFLGAN-TS, C-CosciGAN, and V-CosciGAN show similar performance on the six-feature Sine dataset.

Takeaway: AWD for amplitude features primarily assesses distribution similarity along the temporal dimension. In the two-feature Sine dataset, C-CosciGAN may focus excessively on feature correlations, as illustrated in Fig. 3c, which hampers its ability to capture temporal correlations accurately. On the contrary, V-Coscigan exhibits the best performance w.r.t. AWD since it solely focuses on the temporal dimension and does not need to learn the correlation between the two sine signals, as shown in Fig. 3b. In contrast, VFLGAN-TS achieves the best trade-off.

After training, we select the parameters that achieve minimal AWD in Fig. 2 as the final model to generate synthetic datasets. We use the amplitude in Fig. 2 to estimate the temporal similarity. We now estimate the temporal similarity by the mean absolute error (MAE). For a given synthetic sample ($\tilde{F}_m$), we first estimate the amplitude vector a according to (27). According to the construction of Sine datasets, the ground truth of feature i

Table 1. MAE between Synthetic Datasets and Real Datasets

Dataset	V-CosciGAN	C-CosciGAN	VFLGAN-TS
Two-feature Sine	**0.046**	0.215	0.049
Six-feature Sine	**0.050**	**0.050**	0.051

Table 2. Performance on Downstream Classification Task

	TRTR		TSTS		TRTS		TSTR		TPD	
Method	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**
V-CosciGAN	0.92	0.92	1.00	1.00	0.68	0.68	0.68	0.69	0.56	0.55
C-CosciGAN	0.92	0.92	1.00	1.00	0.87	0.87	0.74	0.75	**0.31**	**0.30**
VFLGAN [60]	0.92	0.92	1.00	1.00	0.85	0.84	0.68	0.70	0.39	0.38
VFLGAN-TS	0.92	0.92	1.00	1.00	0.86	0.86	0.74	0.74	0.32	0.32

TRTR: Train on real test on real; **TSTS**: train on synthetic test on synthetic; **TRTS**: train on real test on synthetic; **TSTR**: train on synthetic test on real; **Ac**: accuracy; **F1**: F1-score; **TPD**: Total performance difference is the final comparison metric and **lower is better**.

(F_{mi}) can be estimated as $F_{mi} = a_i sin(2\pi f_i \boldsymbol{t})$. Then, the MAE can be calculated by,

$$\text{MAE} = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{t=1}^{T} |\tilde{F}_{mn}^{t} - F_{mn}^{t}|. \tag{29}$$

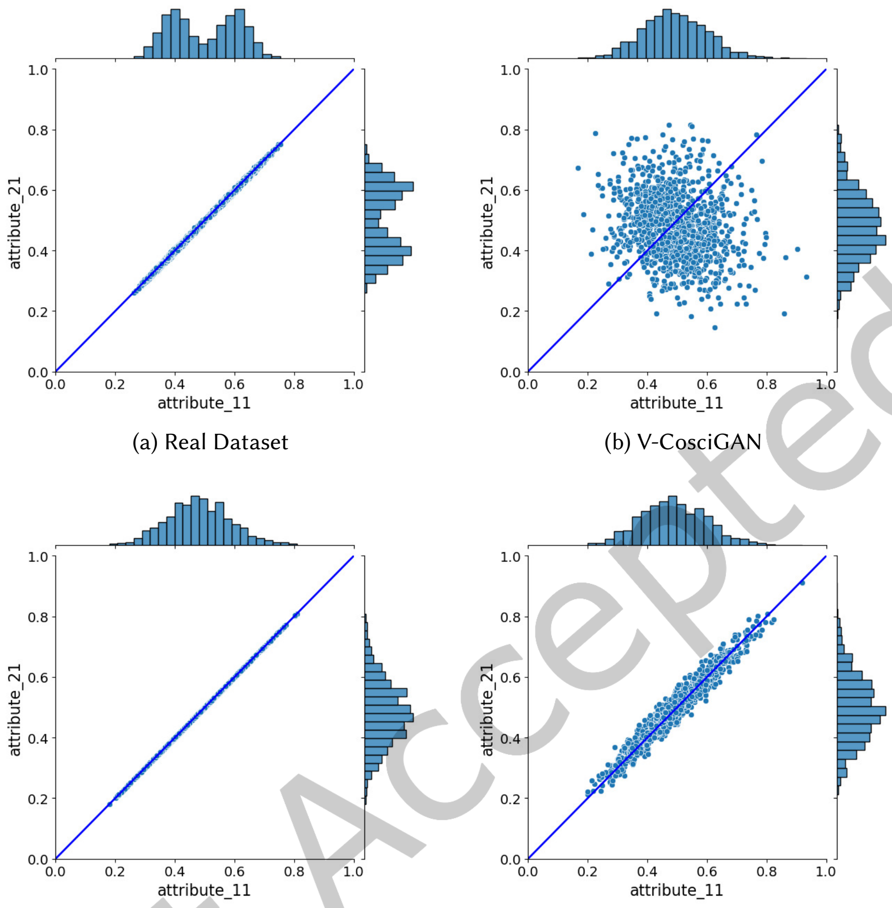
As shown in Table 1, for the two-feature Sine dataset, V-CosciGAN achieves the best performance while C-CoscigGAN presents the worst performance, and for the six-feature Sine dataset, the three methods present similar performance. The MAE and AWD give consistent experimental results, which shows that AWD is an effective metric for measuring temporal similarity. In the case of real-world datasets, MAE is impossible to estimate. Thus, we apply AWD in the following experiments to select the best parameters.

Figure 3 shows the distribution of feature amplitude and the ratio between the amplitudes of feature 1 and feature 2. As shown in Fig. 3a, the ratio between the amplitudes of feature 1 and feature 2 should be around 1.0 for the samples of the real dataset. As shown in Fig. 3c, C-CosciGAN learns this property perfectly. As shown in Fig. 3d, VFLGAN-TS can also learn this property, but the ratio has a larger range compared to C-CosciGAN due to the information loss in the vertically partitioned scenario. On the other hand, VFLGAN-TS achieves significant improvement compared to V-CosciGAN, which cannot learn the correlation of the features located in different parties.

Takeaway: From the evaluation results on Sine datasets, we can conclude that C-CosciGAN performs better in learning the correlation between different features but may perform poorly in learning temporal distribution. On the other hand, VFLGAN-TS performs better in learning the temporal distribution compared to C-CosciGAN and better in learning feature correlation than V-CosciGAN.

5.2.2 EEG Dataset. For the EEG dataset, we first use AWD (23) to evaluate the similarity of each attribute at each time step between real and synthetic datasets. As shown in Fig. 4, the three methods perform similarly in learning the temporal distribution. However, VFLGAN [60] performs poorly and cannot even converge for EEG 1.

Samples in the EEG dataset do not have an explicit construction rule as Sine datasets. Thus, we can not construct the ground truth of the synthetic samples or compute MAE as (29). Instead, we use Performance on Downstream Tasks to evaluate the distribution similarity between synthetic and real datasets and the potential for real-world application. The downstream task for the EGG dataset is classification since the dataset was originally constructed for classification. First, we choose the parameters that achieve minimal AWD to generate synthetic



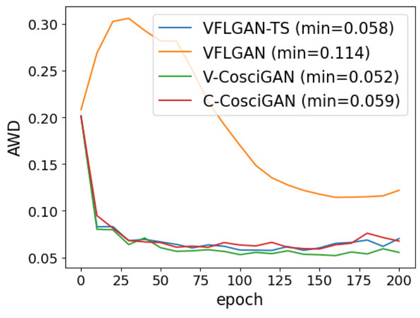
(a) Real Dataset (b) V-CosciGAN

(c) C-CosciGAN (d) VFLGAN-TS

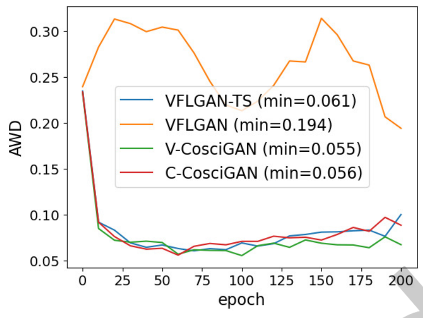
Fig. 3. The histograms in each sub-figure show the amplitude distribution of each attribute. The spots in each sub-figure represent a sample's amplitudes of both attributes.

datasets with the same number of samples as the real dataset. Then, we split all the datasets into training and test sets. Last, we train deep learning models on each training set and then test the models' performance on test sets. We repeated the experiments ten times, and the averages of the performances are presented in Table 2. As shown in Table 2, VFLGAN-TS performs similarly to C-CosciGAN while significantly outperforming V-CosciGAN. The superior performance of VFLGAN-TS and C-CosciGAN comes from the fact that they can learn the feature correlation better since the three methods perform similarly in learning the distribution of each attribute at each time step, as shown in Fig. 4. On the other hand, VFLGAN-TS outperforms VFLGAN since VFLGAN-TS can handle time series data more properly.

Lastly, we employ t-SNE and PCA to visualise the sample distribution, providing an intuitive depiction of the similarity between real and synthetic datasets. Figure 5 shows the distribution similarity between the real and synthetic datasets. According to the t-SNE visualization in Fig. 5, although the performance of the three methods

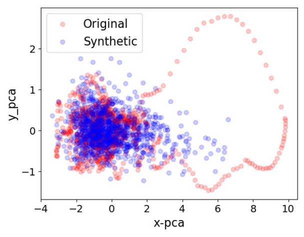


(a) EEG 0

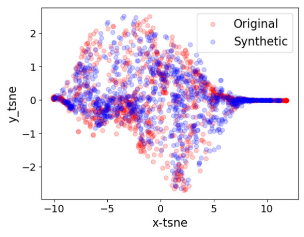


(b) EEG 1

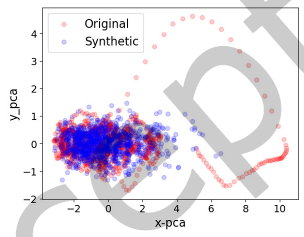
Fig. 4. Wasserstein distance curves during training different methods on EEG Dataset.



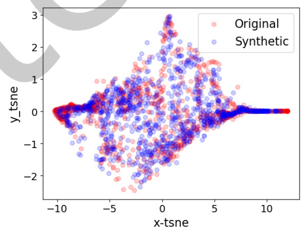
(a) V-CosciGAN (PCA)



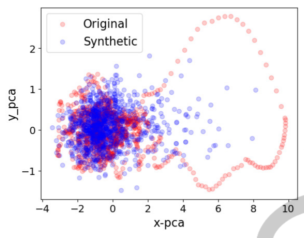
(b) V-CosciGAN (t-SNE)



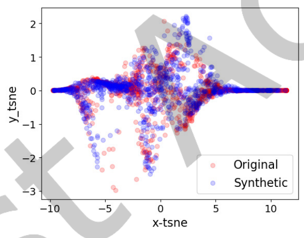
(c) V-CosciGAN (PCA)



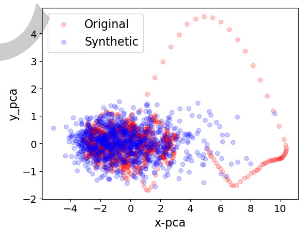
(d) V-CosciGAN (t-SNE)



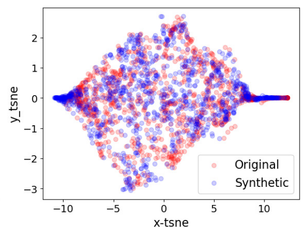
(e) C-CosciGAN (PCA)



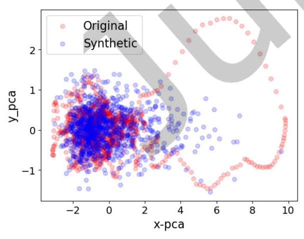
(f) C-CosciGAN (t-SNE)



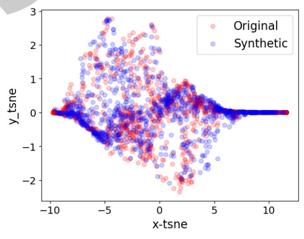
(g) C-CosciGAN (PCA)



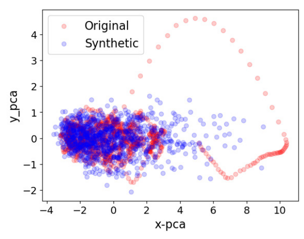
(h) C-CosciGAN (t-SNE)



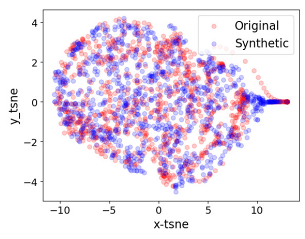
(i) VFLGAN-TS (PCA)



(j) VFLGAN-TS (t-SNE)

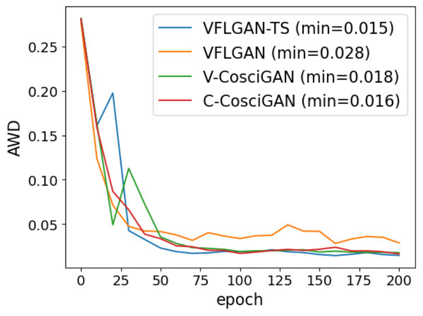


(k) VFLGAN-TS (PCA)

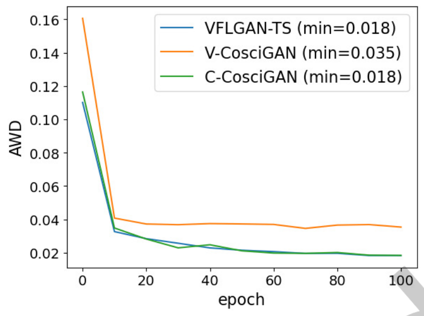


(l) VFLGAN-TS (t-SNE)

Fig. 5. Visualization of similarity between real and synthetic datasets using PCA and t-SNE. The left two columns are results for EEG 0 and the right two columns are results for EEG 1.



(a) Stock



(b) Energy

Fig. 6. Wasserstein distance curves during training different methods on Stock and Energy Datasets. The AWD curve of VFLGAN is not shown in Fig. 6b since the AWD increased to above 2.0 (epoch 100) from 0.33 (epoch 0).

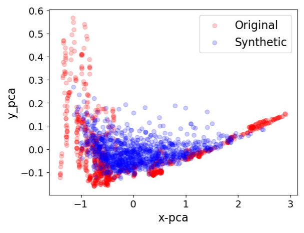
Table 3. Mean Absolute Error (MAE) on Downstream Forecasting Task

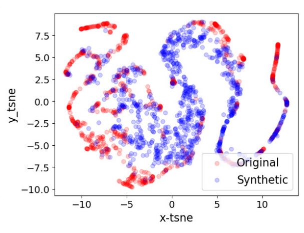
Dataset	Method	TRTR	TSTS	TRTS	TSTR	TPD
Stock	**V-CosciGAN**	0.050 (0.005)	0.042 (0.001)	0.063 (0.005)	0.054 (0.002)	0.025
	C-CosciGAN	0.050 (0.005)	0.048 (0.005)	0.057 (0.006)	0.056 (0.005)	0.015
	VFLGAN [60]	0.050 (0.005)	0.042 (0.004)	0.046 (0.002)	0.053 (0.004)	0.015
	VFLGAN-TS	0.050 (0.005)	0.048 (0.003)	0.050 (0.005)	0.050 (0.003)	**0.002**
Energy	**V-CosciGAN**	0.062 (0.0027)	0.063 (0.0034)	0.122 (0.0025)	0.099 (0.0052)	0.098
	C-CosciGAN	0.062 (0.0027)	0.061 (0.0025)	0.064 (0.0026)	0.066 (0.0020)	**0.007**
	VFLGAN [60]	Not available since VFLGAN did not converge				
	VFLGAN-TS	0.062 (0.0027)	0.060 (0.0034)	0.061 (0.0028)	0.067 (0.0032)	0.008

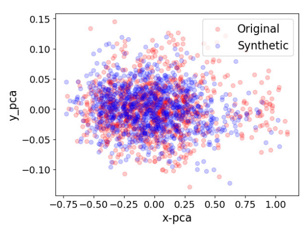
Abbreviations are the same as Table 2. Experiments repeat ten times. Average MAE is shown in this table and standard deviation is shown in brackets.

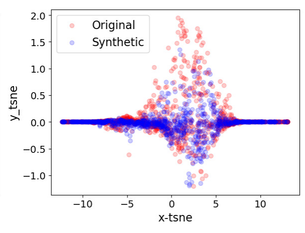
is similar, VFLGAN-TS is slightly better than the other two methods. According to the PCA visualization in Fig. 5, none of the methods satisfactorily capture the distribution of outlier samples, but VFLGAN-TS performs similarly to C-CosciGAN, which is the upper limit for vertically partitioned methods.

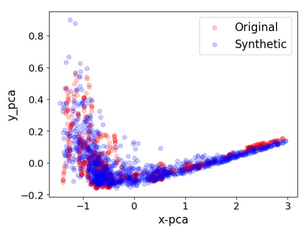
5.2.3 Stock and Energy Datasets. Stock and Energy datasets are real-world datasets, and the time-series length is 24. For these two datasets, we first present the AWD (23) curves in Fig. 6. As shown in Fig. 6a, the three CosciGAN-based methods perform similarly on the Stock dataset while outperforming VFLGAN. As shown in Fig. 6b, VFLGAN-TS and C-CosciGAN outperform V-CosciGAN by a significant margin, and VFLGAN cannot converge on the Energy dataset. Based on Fig. 6b, we can conclude that learning the attribute correlation, i.e., the shared discriminator in VFLGAN-TS and central discriminator in C-CosciGAN, can be beneficial to learning the temporal correlation of within a single attribute. The downstream task for Stock and Energy datasets is forecasting, i.e., forecasting the attribute value of the next time step based on the values of the past 23 time steps. First, we choose the parameters that achieve minimal AWD to generate synthetic datasets with the same number of samples as the real datasets. Then, we split all the datasets into training and test sets. Last, we train deep learning models on each training set and then test the models' performance on test sets. We repeated the experiments ten times, and the average and standard deviation of the performances are presented in Table 3.

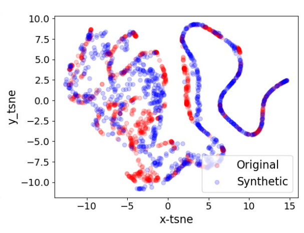

(a) V-CosciGAN (PCA)

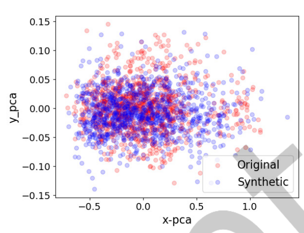

(b) V-CosciGAN (t-SNE)

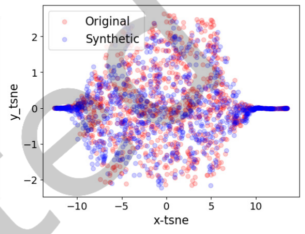

(c) V-CosciGAN (PCA)

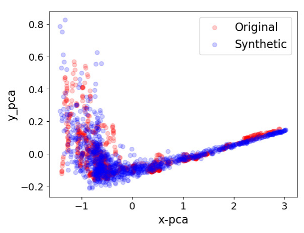

(d) V-CosciGAN (t-SNE)

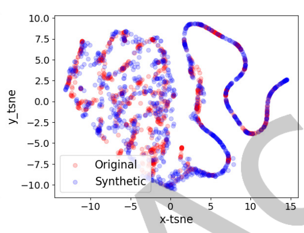

(e) C-CosciGAN (PCA)

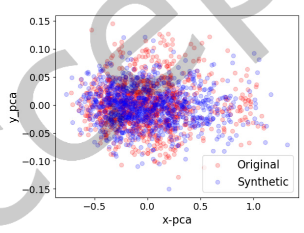

(f) C-CosciGAN (t-SNE)


(g) C-CosciGAN (PCA)


(h) C-CosciGAN (t-SNE)


(i) VFLGAN-TS (PCA)


(j) VFLGAN-TS (t-SNE)


(k) VFLGAN-TS (PCA)

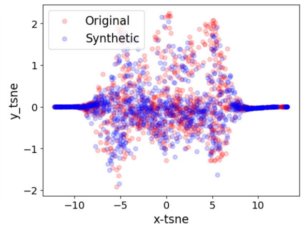

(l) VFLGAN-TS (t-SNE)

Fig. 7. Visualization of similarity between real and synthetic datasets using PCA and t-SNE. The left two columns are results for the Stock dataset and the right two columns are results for the Energy dataset.

VFLGAN-TS and C-CosciGAN outperform V-CosciGAN in both Stock and Energy datasets, as indicated in Table 3. Furthermore, VFLGAN-TS performs superior to C-CosciGAN and VFLGAN on the Stock dataset. Lastly, we employ t-SNE and PCA to visualise the sample distribution in Fig. 7, providing an intuitive depiction of the similarity between real and synthetic samples. According to the left two columns of Fig. 7, the distribution of samples generated by VFLGAN-TS most closely matches that of the real samples, followed by C-CosciGAN and V-CosciGAN, in sequence. According to the right two columns of Fig. 7, VFLGAN-TS and C-CosciGAN present similar performance and outperform V-CosciGAN.

5.2.4 Evaluation of AADA. We apply AADA to evaluate the performance on learning correlation among different time steps within each attribute that is a time series. Figure 8 shows the AADAs on the four datasets, i.e., Stock, Energy, EEG_0, and EGG_1. As shown in Fig. 8, the proposed VFLGAN-TS achieves a similar performance to the centralize C-CosciGAN, indicating that the proposed VFL framework can achieve a minor performance loss due to the vertically partitioned setting. When the attribute set is small (i.e., EEG_0 and EEG_1), V-CosciGAN can also achieve a good performance w.r.t. the AADA due to the effectiveness of attribute generator and discriminator to

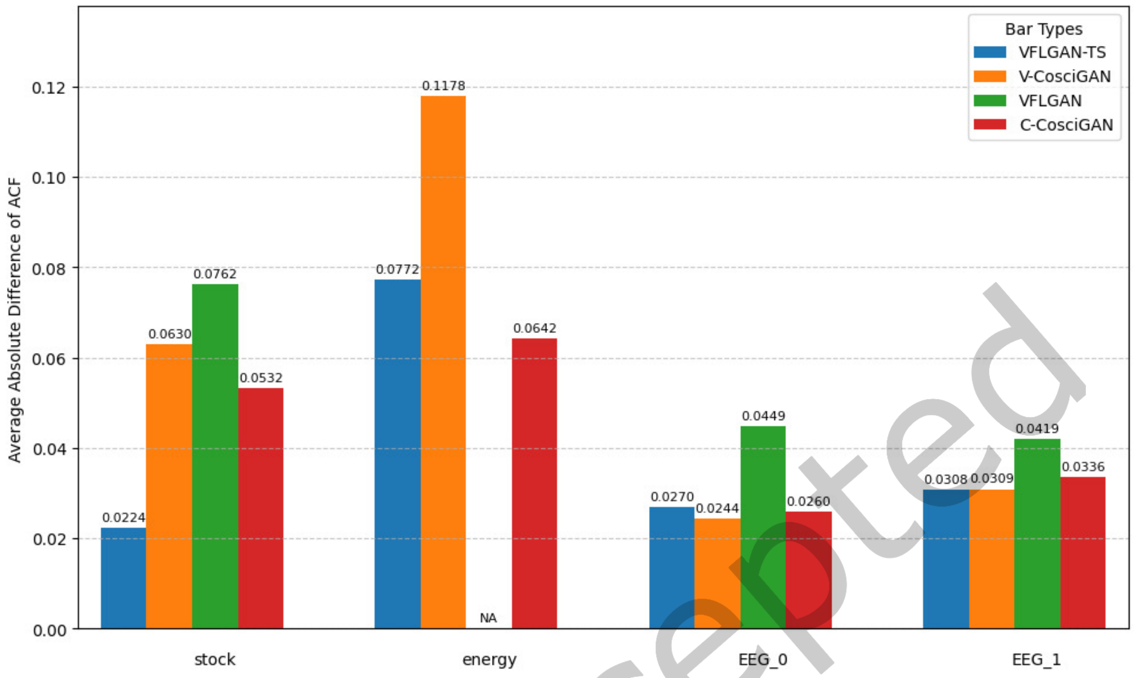


Fig. 8. Bar plot of the Average Absolute Difference of ACF (AADA).

learn the joint distribution among different time steps within time series. On the other hand, when the attribute set is large (e.g., Stock and Energy), VFLGAN-TS and C-CosciGAN perform better, as learning inter-attribute correlations helps improve intra-attribute learning performance if the attributes are correlated. In contrast, VFLGAN perform worst and does not even converge when training on the Energy dataset. The above results are aligned with the AWD curves in Fig. 4 and Fig. 6.

5.2.5 Evaluation of DPVFLGAN-TS with Various DP Budgets. We evaluate the performance of DPVFLGAN-TS with various DP budgets w.r.t. AWD, AADA, and downstream tasks on Stock and Energy datasets. The AWD curves during the training of DPVFLGAN-TS are shown in Fig. 9. As observed, larger DP budgets generally lead to better AWD performance, since a higher budget corresponds to lower variance in the Gaussian noise added to the gradient updates. Moreover, since the Energy dataset is substantially larger than the Stock dataset, the variation in Gaussian noise variances across different DP budgets is less significant for the Energy dataset. Consequently, the AWD curves for different DP budgets are closer when training on Energy than on Stock. Besides, a larger variance in the Gaussian noise (i.e., a smaller DP budget) can result in higher gradient amplitudes, which in turn lead to faster convergence as shown in Fig. 9a. The AADA and downstream task performance of DPVFLGAN-TS with various DP budgets are shown in Fig. 10 and Table 4. These results indicate a similar conclusion to AWD curves, i.e., larger DP budget leads to a better generative quality. In addition, compared to the performance of VFLGAN-TS, providing a DP guarantee will decrease the generative performance, which is a common trade-off between the performance and privacy guarantee as observed in papers such as [6, 22, 57, 61]. The DP budget (ϵ) of VFLGAN-TS is ∞, so $\epsilon = 10$ is a significantly tighter budget compared to the non-DP VFLGAN-TS. As a result, we can observe a noticeable performance decrease for $\epsilon = \{2, 4, 6, 8, 10\}$. On the other hand, a very

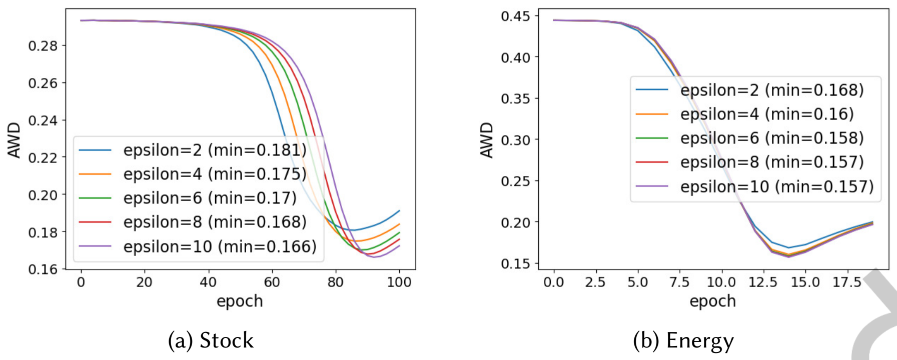


Fig. 9. Wasserstein distance curves during training DPVFLGAN-TS with various DP budgets on Stock and Energy Datasets.

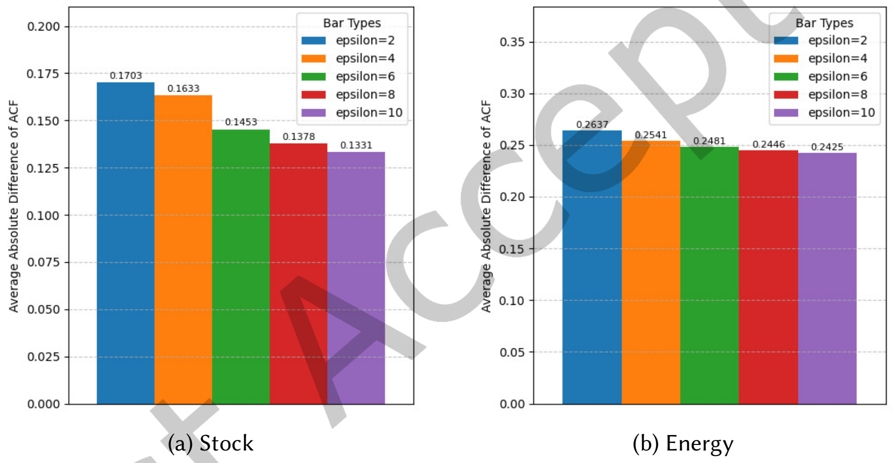
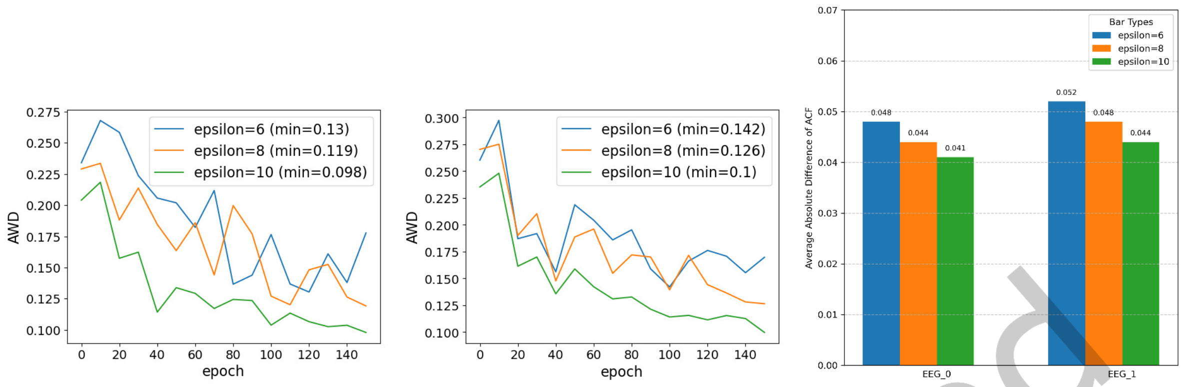


Fig. 10. Bar plot of the Average Absolute Difference of ACF (AADA) where the DPVFLGAN-TS with various DP budgets is trained on Stock and Energy Datasets.

large DP budget (e.g., $\epsilon = 10^9$) is meaningless in theory, although it can lead to a good generative performance. This dilemma is the key reason why we need a strong MIA for privacy auditing. As shown in Table 9, although $\epsilon = 10$ is theoretically a loose privacy budget, the resulting privacy leakage is negligible. This is because the DP calculates for the worst-case privacy risk, which usually does not exist in real-world datasets. Thus, a strong MIA can help us to audit the real risk of privacy breaches, and we can choose a large ϵ to achieve a balance between performance and practical privacy risk. We also evaluate DPVFLGAN-TS on EEG datasets. However, due to the small size of EEG datasets, the variance of the Gaussian noise is large for a tight DP budget. As a result, DPVFLGAN-TS cannot converge when $\epsilon = \{2, 4\}$. The AWD curves and AADA bars are shown in Fig. 11. The conclusion is the same as training DPVFLGAN-TS on Stock and Energy datasets, i.e., a larger DP budget leads to better generative quality.



(a) EEG_0 (b) EEG_1 (c) AADA

Fig. 11. Wasserstein distance curves and AADA bars when training DPVFLGAN-TS on EEG datasets.

Table 4. Mean Absolute Error (MAE) on Downstream Forecasting Task of DPVFLGAN-TS with various DP budgets

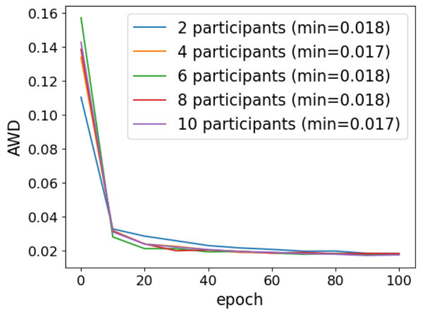
Dataset	DP budgets	TRTR	TSTS	TRTS	TSTR	TPD
Stock	$(2, 3\times10^{-4})$	0.050	0.023	0.079	0.126	0.132
	$(4, 3\times10^{-4})$	0.050	0.023	0.066	0.124	0.117
	$(6, 3\times10^{-4})$	0.050	0.023	0.062	0.115	0.104
	$(8, 3\times10^{-4})$	0.050	0.022	0.060	0.111	0.099
	$(10, 3\times10^{-4})$	0.050	0.022	0.057	0.109	0.094
Energy	$(2, 5\times10^{-5})$	0.062	0.024	0.124	0.186	0.224
	$(4, 5\times10^{-5})$	0.062	0.024	0.112	0.178	0.204
	$(6, 5\times10^{-5})$	0.062	0.023	0.109	0.175	0.199
	$(8, 5\times10^{-5})$	0.062	0.023	0.105	0.173	0.193
	$(10, 5\times10^{-5})$	0.062	0.023	0.101	0.169	0.185

Abbreviations are the same as Table 2. Experiments repeat ten times and average MAE is shown in this table.

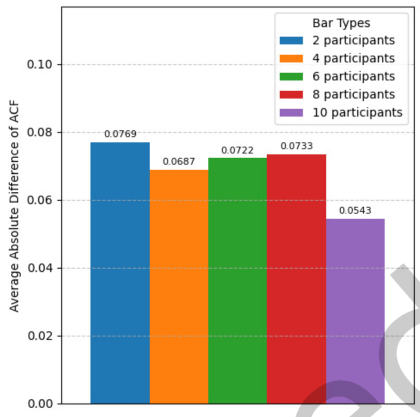
5.2.6 Evaluation of VFLGAN-TS on More Participants. In previous sections, we evaluated VFLGAN-TS in two-participant scenarios. In this subsection, we assess its robustness in {2, 4, 6, 8, 10}-participant settings using the Energy dataset, as the other datasets lack sufficient attributes to support scenarios with 8 or 10 participants. We evenly distribute the attributes among participants as much as possible. Figure 12 shows the AWD curves during training and AADA bar plot of VFLGAN-TS for {2, 4, 6, 8, 10}-participant scenarios on Energy Datasets. According to the AWD and AADA bar plot, increasing the number of participants has a limited impact on the generative capability of VFLGAN-TS. Table 5 shows the MAE of the downstream task where VFLGAN-TS is trained in {2, 4, 6, 8, 10}-participant scenarios. Aligned with AWD and AADA, the number of participants has a limited impact on the generation capability of VFLGAN-TS.

5.3 Analysis of Computation and Time Complexity

This section analyzes the computational and time complexity of VFLGAN-TS when trained on the EEG, Energy, and Stock datasets, as the model architecture varies slightly across these datasets. We focus on the two-participant



(a) AWD



(b) AADA

Fig. 12. AWD curves and AADA bar plot of VFLGAN-TS for {2, 4, 6, 8, 10}-participant scenarios on Energy Datasets.

Table 5. Mean Absolute Error (MAE) on Downstream Forecasting Task of VFLGAN-TS with various Participants

Dataset	Participant Number	TRTR	TSTS	TRTS	TSTR	TPD
Energy	2	0.062	0.058	0.060	0.065	0.009
	4	0.062	0.061	0.063	0.066	0.006
	6	0.062	0.061	0.062	0.065	0.004
	8	0.062	0.062	0.066	0.066	0.008
	10	0.062	0.059	0.061	0.065	0.007

Abbreviations are the same as Table 2. Experiments repeat ten times and average MAE is shown in this table.

case, as increasing the number of participants reduces the number of attributes per participant and lowers resource requirements. The number of floating-point operations (FLOPs) is a standard and hardware-independent metric to quantify the computational complexity of deep learning models. We analyze the FLOPs per forward pass to assess the computation complexity of the VFLGAN-TS. We omit latency, as it depends on the participants' network capabilities. Additionally, the reported training time excludes communication latency, i.e., ideal training time (ITT) with infinite bandwidth. The practical training time can be estimated by combining the ITT with the total communication time, which can be calculated by the total communication volume and a given participants' bandwidth. Table 6 summarizes information about computation, memory, and communication volume in the participant maintaining the most attributes among all participants. As shown in Table 6, datasets with more attributes (e.g., Energy) and longer time segmentation T (e.g., EEG) result in higher FLOPs and size of parameters (SoP). In the field of deep learning, VFLGAN-TS is a tiny model w.r.t. the FLOPs and SoP shown in Table 6. Regardless of the dataset, communication volume per round (CV_r) is fixed due to the fixed output size of the Feature Extractor. The total communication volume (CV_t) depends on both CV_r and total training rounds. Despite the large difference in FLOPs between the VFLGAN-TSs trained on the Energy and Stock datasets, their memory usage remains similar, as shown in Table 6. This is because the local attribute generators and discriminators are executed sequentially, which also explains the significant difference in ITTs when training

Table 6. Computation, Memory, and Communication Volume per Participants

Dataset	FLOPs	Memory	SoP	CV_r	CV_T	ITT
Energy	289.75M	731MB	17.98MB	1MB	30.8GB	3656s
Stock	61.95M	621MB	3.86MB	1MB	11.6GB	346s
EEG_0&1	71.28M	601MB	4.44MB	1MB	3.2GB	83s

SoP: size of parameters; CV_r: communication volume per round; CV_t: total communication volume; ITT: ideal training time. M = 10^6; G = 10^9.

VFLGAN-TS on the Energy and Stock datasets, where the Energy dataset has many more attributes than the Stock dataset.

5.4 Experimental Environment and Kay Parameters

All the experiments are conducted in a Linux server with 256 CPUs (AMD EPYC 7742 64-Core Processor) and 8 GPUs (NVIDIA A100-SXM4-80GB), which is shared by multiple users. In the experiments, we only activated one out of eight GPUs since our model is tiny. We use Anaconda to manage our Python (3.10) environment, with PyTorch 2.3.1 as the core library. The local attribute discriminator consists of three MLP layers: the first two are followed by LeakyReLU activations, and the final MLP layer is followed by a Sigmoid function. The dimensions [input, output] of the three MLP layers are: [T, 128], [128, 64], and [64, 1], where T is the length of the time series. The local feature extractor consists of one MLP layer followed by LeakyReLU. The dimension [input, output] of the MLP layer is [$T \times |A_i|$, 256], where $|A_i|$ is the number of local attributes of Participant i. The local attribute generator consists of a 1-layer LSTM layer and an MLP layer in sequence. The latent dimension and hidden dimension of the LSTM layer is 32 and 256, respectively. The dimension [input, output] of the MLP layer is [32, T]. The shared discriminator also consists of three MLP layers: the first two are followed by LeakyReLU activations, and the final MLP layer is followed by a Sigmoid function. The dimensions [input, output] of the three MLP layers are: [$256 \times M$, 256], [256, 64], and [64, 1], where M is the number of participants. We use Adam optimizer for training, where the learning rate for attribute discriminators and generators is 0.0002 and the learning rate for feature extractor and shared discriminator is 0.0001. The betas for all optimizers are [0.5, 0.9]. The training epochs for EEG_0, EEG_1, Stock, and Energy datasets are 200, 200, 200, and 100, respectively. The mini-batch size for all experiments is 64.

5.5 Privacy Analysis

In this section, we apply the enhanced ASSD to analyze the privacy breaches through synthetic datasets. Additionally, we apply the Wilcoxon signed-rank test to calculate the p-value comparing the AUCs of ASSD and enhanced ASSD. The null hypothesis (H_0) states that the median of the differences (between the AUCs of ASSD and enhanced ASSD) is zero, while the alternative hypothesis (H_1) asserts that the median difference is non-zero, indicating a significant difference. First, we use ASSD to identify two vulnerable records to attack. While the original ASSD leverages naive and correlation features, we additionally apply the proposed KNN feature to attack the selected records, demonstrating its superiority. We compute the p-values between AUC of KNN feature and AUC of {Naive, Corr} features, and then record the maximum of the two in Table 7. Table 7 shows the mean AUC of attacking each record. As shown in Table 7, attacking with KNN feature usually leads to a higher mean AUC compared to the naive and correlation features. Next, we select the vulnerable record with the enhanced ASSD and attack the record with naive, correlation, and KNN features. Table 8 shows the mean AUC of attacking the vulnerable record selected by ASSD with naive, correlation, and KNN features. **Remark**: the AUC of the KNN feature in Table 8 is the performance of the enhanced ASSD, i.e., the vulnerable record is selected by the

Table 7. Mean AUC of ASSD

	Naive	Corr	KNN	p-value	Naive	Corr	KNN	p-value
Dataset		**Record 1**				**Record 2**		
EEG 0	0.53	0.52	0.55	0.375	0.52	0.49	0.52	1.0
EEG 1	0.49	0.43	0.53	0.083	0.52	0.45	0.57	0.073
Stock	0.48	0.51	0.55	0.038	0.50	0.50	0.56	0.004
Energy	0.50	0.51	0.58	0.020	0.51	0.48	0.55	0.027

Record 1 and Record 2 is selected by ASSD. We repeat all experiments ten times and calculate the mean AUC. We apply the proposed KNN feature to attack records 1 and 2 to show its superiority. We compute the p-values between the AUC of the KNN feature and the AUC of {Naive, Corr} features, and then record the maximum of the two in this table.

Table 8. Mean AUC of enhanced ASSD

Dataset	Naive	Corr	KNN	p-value
EEG 0	0.54	0.53	0.59	0.037
EEG 1	0.48	0.51	0.58	0.048
Stock	0.50	0.52	0.60	0.002
Energy	0.82	0.86	0.59	0.012

The attack record is selected by the enhanced ASSD. We repeat all experiments ten times and calculate the mean AUC. We compute p-values between the AUC of the KNN feature in this table and the AUC of {Naive, Corr} features of the two records in Table 7, and then record the maximum one.

enhanced ASSD and is attacked with the proposed KNN feature. As a result, we compute p-values between the AUC of the KNN feature in Table 8 and the AUC of {Naive, Corr} features of the two records in Table 7, and then record the maximum one, to demonstrate the statistical significance between the enhanced ASSD and the original ASSD. The p-values in Table 8 are extremely small, indicating that the proposed enhanced ASSD achieves a better attacking performance than the original ASSD. We then evaluate the robustness of DPVFLGN-TS under multiple DP budgets against the MIA attacks. As shown in Table 9, the privacy breaches become negligible with all DP budgets. **Remark**: the MIA fails when AUC ≤ 0.5. To satisfy a tight DP guarantee, i.e., $\epsilon = \{2, 4\}$, excessive Gaussian noise needs to be added to the gradients when training on EEG datasets due to their small size, which causes the model not to converge. On the other hand, as shown in Table 9, $\epsilon = 10$ is robust enough against the MIA attack. Lastly, we utilise the ASIF as proposed in [60] to assess the breach of privacy through the features transmitted to the server during the training process. As shown in Table 10, although the privacy breach of the targets in Table 8 is significant through synthetic datasets, the privacy breach is negligible through the features, even for the non-DP method.

6 Conclusion and Discussion

This paper proposed VFLGAN-TS, which is the *first* generative model for publishing time-series data in vertically partitioned scenarios. The differentially private VFLGAN-TS is provided to reduce privacy breaches. Then, we enhanced the privacy auditing scheme that we previously proposed for auditing VFLGAN to evaluate the privacy breach of synthetic time-series data. The experimental results show that the performance of VFLGAN-TS is close to its upper limit, i.e., C-CosciGAN trained in a centralized manner, and DPVFLGAN-TS can significantly reduce privacy breaches. One limitation of this paper is that neither centralised CosciGAN nor VFLGAN-TS can effectively capture the distribution of outlier samples, which is why the privacy leakage of some outliers is negligible. Future research could focus on enhancing the model's ability to learn the distribution of inliers and

Table 9. Mean AUC of enhanced ASSD when Attacking DPVFLGAN-TS

	Naive	**Corr**	**KNN**	**Naive**	**Corr**	**KNN**	**Naive**	**Corr**	**KNN**	**Naive**	**Corr**	**KNN**
DP (ϵ)	**EEG_0**			**EEG_1**			**Stock**			**Energy**		
2	NA	NA	NA	NA	NA	NA	0.45	0.41	0.41	0.52	0.42	0.45
4	NA	NA	NA	NA	NA	NA	0.45	0.46	0.52	0.44	0.43	0.44
6	0.49	0.46	0.43	0.49	0.44	0.5	0.48	0.50	0.45	0.43	0.52	0.48
8	0.48	0.45	0.46	0.47	0.49	0.52	0.50	0.45	0.51	0.50	0.43	0.44
10	0.49	0.51	0.52	0.49	0.45	0.51	0.49	0.49	0.53	0.50	0.48	0.49

The selected record is the same as Table 8. δ is 10^{-3}, 3×10^{-4}, and 5×10^{-5}, for EEG, Stock, and Energy datasets, respectively, since it depends on the size of the dataset. NA: The model cannot converge due to the excessive noise.

Table 10. Mean AUC of ASIF through Features

Dataset	Method	**Naive**		**Corr**	
		F_1	F_2	F_1	F_2
EEG 0	VFLGAN-TS	0.55	0.53	0.46	0.55
	$(10, 10^{-3})$-DP	0.49	0.54	0.49	0.54
EEG 1	VFLGAN-TS	0.48	0.47	0.53	0.46
	$(10, 10^{-3})$-DP	0.48	0.52	0.50	0.48
Stock	VFLGAN-TS	0.50	0.51	0.52	0.51
	$(10, 3 \times 10^{-4})$-DP	0.50	0.48	0.51	0.50
Energy	VFLGAN-TS	0.49	0.47	0.46	0.49
	$(10, 5 \times 10^{-5})$-DP	0.54	0.54	0.50	0,52

The targets are the same as Table 8; F_1 and F_2 are the features from the feature extractor of Party 1 and Party 2, respectively.

outliers. Furthermore, although this paper has made an effort to select vulnerable samples, determining the most vulnerable sample for privacy analysis remains an under-explored research topic.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Ahmed Alaa, Alex James Chan, and Mihaela van der Schaar. 2021. Generative time-series modeling with fourier flows. In *International Conference on Learning Representations*.

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

[4] Fred B Bryant and Paul R Yarnold. 1995. Principal-components analysis and exploratory and confirmatory factor analysis. (1995).

[5] Luis Candanedo. 2017. Appliances Energy Prediction. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VC8G.

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

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

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

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

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

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

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

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

[14] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633* (2017).

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

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

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

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

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

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

[21] Jinsung Jeon, Jeonghak Kim, Haryong Song, Seunghyeon Cho, and Noseong Park. 2022. GT-GAN: General purpose time series synthesis with generative adversarial networks. *Advances in Neural Information Processing Systems* 35 (2022), 36999–37010.

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

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

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

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

[26] Jintao Liang, Sen Su, and Zhenya Wang. 2025. Vertical Federated Representation Synthesis for Non-aligned Samples in the Active Party. *IEEE Transactions on Big Data* (2025), 1–15. doi:10.1109/TBDATA.2025.3566595

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

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

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

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

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

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

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

[34] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).

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

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

[37] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. 2018. T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *arXiv preprint arXiv:1811.08295* (2018).

[38] Oliver Roesler. 2013. EEG Eye State. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C57G7J.

[39] Ali Seyfi, Jean-Francois Rajotte, and Raymond Ng. 2022. Generating multivariate time series with COmmon Source CoordInated GAN (COSCI-GAN). *Advances in Neural Information Processing Systems* 35 (2022), 32777–32788.
[40] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.
[41] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic Data - Anonymisation Groundhog Day. In *USENIX Security Symposium*.
[42] Thomas Steinke, Milad Nasr, and Matthew Jagielski. 2024. Privacy auditing with one (1) training run. *Advances in Neural Information Processing Systems* 36 (2024).
[43] Peng Tang, Xiang Cheng, Sen Su, Rui Chen, and Huaxi Shao. 2021. Differentially Private Publication of Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 18, 2 (2021), 780–795. doi:10.1109/TDSC.2019.2905237
[44] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2* (2019).
[45] Boris van Breugel, Hao Sun, Zhaozhi Qian, and Mihaela van der Schaar. 2023. Membership inference attacks against synthetic data through overfitting detection. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
[46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[47] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.
[48] Shuo Wang, Keke Gai, Jing Yu, Zijian Zhang, and Liehuang Zhu. 2025. PraVFed: Practical Heterogeneous Vertical Federated Learning via Representation Learning. *IEEE Transactions on Information Forensics and Security* 20 (2025), 2693–2705. doi:10.1109/TIFS.2025.3530700
[49] Xiaoding Wang, Sahil Garg, Hui Lin, Jia Hu, Georges Kaddoum, Md. Jalil Piran, and M. Shamim Hossain. 2022. Toward Accurate Anomaly Detection in Industrial Internet of Things Using Hierarchical Federated Learning. *IEEE Internet of Things Journal* 9, 10 (2022), 7110–7119. doi:10.1109/JIOT.2021.3074382
[50] Xiaoding Wang, Sahil Garg, Hui Lin, Georges Kaddoum, Jia Hu, and Mohammad Mehedi Hassan. 2023. Heterogeneous Blockchain and AI-Driven Hierarchical Trust Evaluation for 5G-Enabled Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems* 24, 2 (2023), 2074–2083. doi:10.1109/TITS.2021.3129417
[51] Xiaoding Wang, Sahil Garg, Hui Lin, Georges Kaddoum, Jia Hu, and M. Shamim Hossain. 2022. A Secure Data Aggregation Strategy in Edge Computing and Blockchain-Empowered Internet of Things. *IEEE Internet of Things Journal* 9, 16 (2022), 14237–14246. doi:10.1109/JIOT.2020.3023588
[52] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1226–1235.
[53] Wikipedia. 2024. Wasserstein metric. https://en.wikipedia.org/wiki/Wasserstein_metric
[54] Yunpeng Xiao, Yutong Guo, Haipeng Zhu, Chaolong Jia, Qian Li, Rong Wang, and Guoyin Wang. 2024. An Unsupervised Federated Domain Adaptation Method Based on Knowledge Distillation. *IEEE Transactions on Neural Networks and Learning Systems* (2024), 1–15. doi:10.1109/TNNLS.2024.3510382
[55] Yunpeng Xiao, Xufeng Li, Tun Li, Rong Wang, Yucai Pang, and Guoyin Wang. 2025. A Distributed Generative Adversarial Network for Data Augmentation Under Vertical Federated Learning. *IEEE Transactions on Big Data* 11, 1 (2025), 74–85. doi:10.1109/TBDATA.2024.3375150
[56] Yunpeng Xiao, Dengke Zhao, Xufeng Li, Tun Li, Rong Wang, and Guoyin Wang. 2025. A Federated Learning-based Data Augmentation Method for Privacy Preservation under Heterogeneous Data. *IEEE Transactions on Mobile Computing* (2025).
[57] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739* (2018).
[58] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. 2022. Enhanced Membership Inference Attacks against Machine Learning Models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 3093–3106.
[59] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. 2019. Time-series generative adversarial networks. *Advances in neural information processing systems* 32 (2019).
[60] Xun Yuan, Yang Yang, Prosanta Gope, Aryan Pasikhani, and Biplab Sikdar. 2024. VFLGAN: Vertical Federated Learning-based Generative Adversarial Network for Vertically Partitioned Data Publication. In *Proceedings on Privacy Enhancing Technologies*.
[61] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594* (2018).

A Proof of Optimality and Convergence of VFLGAN-TS

Theorem A.1. *(Optimality of VFLGAN-TS) The global optimal point of VFLGAN-TS training objective is achieved if and only if $P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}$, namely, the distribution of real and synthetic data are identical.*

Proof. Recall the classical GAN's objective in (1). Let the objective be denoted as $V(G, D)$, namely, the GAN's min-max objective be

$$\min_{G} \max_{D} V(G, D) = \min_{G} \max_{D} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim P_{G(\boldsymbol{z})}}[\log(1 - D(\tilde{\boldsymbol{x}}))] \tag{30}$$

Observing VFLGAN-TS's architecture, combination of all attribute generators $G_{ij}, i \in \{1, 2, \ldots, M\}, j \in \{1, 2, \ldots, |A_i|\}$ can be considered a conceptual global generator of the complete multivariate time-series $\tilde{\mathbf{x}}$, and the shared discriminator D_S with the feature extractors $FE_i, i \in \{1, 2, \ldots, M\}$ can be considered a conceptual global discriminator of $\mathbf{x}$ and $\tilde{\mathbf{x}}$. Formally, we can write

$$G(z) = \left((G_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M} = [G_{11}(z), G_{12}(z), \ldots, G_{1|A_1|}(z), G_{21}(z), \ldots, G_{M1}(z), G_{M2}(z), \ldots, G_{M|A_M|}(z)] \tag{31}$$

$$D(\mathbf{x}) = D_S([FE_1(\mathbf{x}_1), FE_2(\mathbf{x}_2), \ldots, FE_M(\mathbf{x}_M)]) \tag{32}$$

Thus, combine (7) and (8) and substitute (30), (31) and (32) in, the global objective can be written as

$$\begin{aligned}
&\min_{\left((G_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}} \max_{\left((D_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}, (FE_i)_{i=1}^{M}, D_S} \left[\sum_{i=1}^{M} \sum_{j=1}^{|A_i|} \underbrace{\Big(\mathbb{E}[\log(D_{ij}(x_{ij}))] + \mathbb{E}[1 - \log(1 - D_{ij}(G_{ij}(z)))]\Big)}_{\text{per-attribute objective from (7)}} \right] \\
&\qquad + \underbrace{\Big(\mathbb{E}[\log(D(\mathbf{x}))] + \mathbb{E}[\log(1 - D(G(z)))]\Big)}_{\text{shared objective from (8)}} \\
&= \min_{\left((G_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}} \max_{\left((D_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}, (FE_i)_{i=1}^{M}, D_S} \left(\sum_{i=1}^{M} \sum_{j=1}^{|A_i|} V(G_{ij}, D_{ij}) \right) + V(G, D)
\end{aligned} \tag{33}$$

Constant positive coefficients β_1, β_2 are omitted in (33) (i.e., equivalent to $\beta_1 = \beta_2 = 1$) so that the global objective can be stated in one equation.

Let $\mathcal{D}_G^{\dagger} = \arg\max_D V(G, D)$ be the set of optimal Ds given G. Proposition 1 in the original GAN paper [15] states that

$$D_G^{\dagger} = \frac{P}{P + P_{G(z)}}, \forall D_G^{\dagger} \in \mathcal{D}_G^{\dagger} \tag{34}$$

Thus, given G (including G_{ij} for all i and j),

$$\begin{aligned}
\max_{\left((D_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}, (FE_i)_{i=1}^{M}, D_S} \left(\sum_{i=1}^{M} \sum_{j=1}^{|A_i|} V(G_{ij}, D_{ij}) \right) + V(G, D) &\le \left(\sum_{i=1}^{M} \sum_{j=1}^{|A_i|} \max_{D_{ij}} V(G_{ij}, D_{ij}) \right) + \max_{D} V(G, D) \\
&= \left(\sum_{i=1}^{M} \sum_{j=1}^{M} V(G_{ij}, D_{ij}{}_{G_{ij}}^{\dagger}) \right) + V(G, D_G^{\dagger}), \forall D_{ij}{}_{G_{ij}}^{\dagger} \in \mathcal{D}_{ij}{}_{G_{ij}}^{\dagger}, D_G^{\dagger} \in \mathcal{D}_G^{\dagger}
\end{aligned} \tag{35}$$

Note that given G, all the discriminators (D_{ij} for all i and j and D) are independent. This means $D_{ij} = D_{ij}{}_{G_{ij}}^{\dagger}, \forall i, j$ and $D = D_G^{\dagger}$ can be achieved simultaneously, making the inequality sign in (35) can achieve equality with a solution as follows:

$$D_{ij} = D_{ij}{}_{G_{ij}}^{\dagger} = \frac{P_{x_{ij}}}{P_{x_{ij}} + P_{\tilde{x}_{ij}}}, \forall i \in \{1, 2, \ldots, M\}, j \in \{1, 2, \ldots, |A_i|\} D = D_G^{\dagger} = \frac{P_{\mathbf{x}}}{P_{\mathbf{x}} + P_{\tilde{\mathbf{x}}}} \tag{36}$$

Also note that Theorem 1 in the original GAN paper [15] states that the global optimal point of the objective in (30) is achieved if and only if $P = P_{G(z)}$. Let the optimal solutions of G be $\mathcal{G}^*$, then $\mathcal{G}^* = \{G^* | P = P_{G^*(z)}\}$. Let $U(G)$ be the objective function when the discriminators optimal given G, so that

$$\min_G U(G) = \min_G \max_D V(G, D) = \min_G V(G, D^\dagger_G); \qquad \arg\min_G U(G) = \mathcal{G}^* \tag{37}$$

Substitute (36) and (37) in (33) and apply the Theorem,

$$\begin{aligned}\min_{\left((G_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}} \left(\sum_{i=1}^{M}\sum_{j=1}^{M} U(G_{ij})\right) + U(G) &= \min_{\left((G_{ij})_{j=1}^{|A_i|}\right)_{i=1}^{M}} \left(\sum_{i=1}^{M}\sum_{j=1}^{M} V(G_{ij}, D_{ij}{}^{\dagger}_{G_{ij}})\right) + V(G, D^\dagger_G)\\ &\geq \left(\sum_{i=1}^{M}\sum_{j=1}^{M} \min_{G_{ij}} V(G_{ij}, D_{ij}{}^{\dagger}_{G_{ij}})\right) + \min_G V(G, D^\dagger_G) = \left(\sum_{i=1}^{M}\sum_{j=1}^{M} V(G^*_{ij}, D_{ij}{}^{\dagger}_{G^*_{ij}})\right) + V(G^*, D^\dagger_{G^*}),\\ &\forall G^*_{ij} \in \mathcal{G}^*_{ij}, G^* \in \mathcal{G}^*, D_{ij}{}^{\dagger}_{G^*_{ij}} \in \mathcal{D}_{ij}{}^{\dagger}_{G^*_{ij}}, D^\dagger_{G^*} \in \mathcal{D}^\dagger_{G^*}\end{aligned} \tag{38}$$

Consider any $G^* \in \mathcal{G}^* = \{G^* | P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}\}$, the generator for j-th attribute of i-th party of G^* must satisfy $G^*_{ij} \in \mathcal{G}^*_{ij} = \{G^*_{ij} | P_{x_{ij}} = P_{\tilde{x}_{ij}}\}$. Therefore, $\mathcal{G}^* = \{G^* | P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}\}$ is a set of optimal solution to (38) that makes the inequality sign achieve equality. This proves that the global optimal point of (33) is achieved if $P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}$.

Then, suppose a G such that at least one of its attribute generators is not in the optimal set (i.e., $\exists G_{ij} \notin \mathcal{G}^*_{ij}$), then the inequality sign of (38) cannot take equality due to G_{ij}, and $P_{\mathbf{x}} \neq P_{\tilde{\mathbf{x}}}$ because $P_{x_{ij}} \neq P_{\tilde{x}_{ij}}$. Also, suppose $G \notin \mathcal{G}^*$, then the inequality sign of (38) cannot take equality either due to G, and by definition of $\mathcal{G}^*$ we must have $P_{\mathbf{x}} \neq P_{\tilde{\mathbf{x}}}$. Given that the inequality sign of (38) can be achieved by $G^* \in \mathcal{G}^*$, both the above scenarios are not optimal solutions. This proves that the global optimal point of (33) is achieved only if $P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}$.

Although some coefficients on the shared objective when updating different networks are ommitted in (33), they do not change the conclusions above because the coefficients are positive constants.

Therefore, the global optimal point of VFLGAN-TS training objective is achieved if and only if $P_{\mathbf{x}} = P_{\tilde{\mathbf{x}}}$. □

Theorem A.2. *(Convergence of VFLGAN-TS) If all VFLGAN-TS networks ($G_{ij}, D_{ij}, FE_i, D_S, \forall i \in \{1, 2, \ldots, M\}, j \in \{1, 2, \ldots, |A_i|\}$) have enough capacity, and at each step the discriminators (i.e., D_{ij}, FE_i, D_S) are allowed to reach their optimum given the (corresponding) generators (G_{ij}, G), and $P_{\tilde{\mathbf{x}}} = P_{G(z)}$ is updated to improve the criterion in* (38) *(LHS), then $P_{\tilde{\mathbf{x}}} = P_{G(z)}$ converges to $P_{\mathbf{x}}$.*

Proof. By the convergence justified in Proposition 2 of the original GAN paper [15], in a classical GAN with one shared generator and one shared discriminator, given the discriminator updated to its optimum at each step and generator updated to improve the criterion with the optimal discriminator, p_g converges to p_{data} (in the notation of our paper, $P_{\tilde{\mathbf{x}}} = P_{G(z)}$ converges to $P_{\mathbf{x}}$.

Thus, if G_{ij} is updated with D_{ij} only (i.e., no shared discriminator D from D_S and FE_i), $P_{\tilde{x}_{ij}} = P_{G_{ij}(z)}$ converges to $P_{x_{ij}}$. Similarly, if G as the combined generator of each G_{ij} is updated with the shared discriminator only, $P_{\tilde{\mathbf{x}}} = P_{G(z)}$ converges to $P_{\mathbf{x}}$.

Note that the convergence tendency of the combined generator G is completely compatible with the convergence tendency of the local attribute generators G_{ij}, as $P_{\tilde{\mathbf{x}}} = P_{G(z)} = P_{\mathbf{x}}$ must yield $P_{\tilde{x}_{ij}} = P_{G_{ij}(z)} = P_{x_{ij}}$. Therefore, the convergence tendency of the combined generator G remains the same with local attribute generators' convergence tendencies. In other words, in VFLGAN-TS, $P_{\tilde{\mathbf{x}}} = P_{G(z)}$ converges to $P_{\mathbf{x}}$. □

B Proof of Differential Privacy of DPVFLGAN-TS

This section will provide proof of Theorem 4.3, presented in the main paper. We need the following two propositions to prove Theorem 4.3, whose numbers follow the proposition number of the main paper.

Proposition 4. *(Gaussian Mechanism) Let $f : D \to R$ be an arbitrary function with sensitivity being*

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

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

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

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

Proof. Proof can be found in [33]. □

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

Proof. Proof can be found in [13]. □

Theorem B.1. *(RDP Guarantee) All the local attribute discriminators and feature extractors satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP and the local attribute generators satisfy $(\alpha, \alpha/\sigma^2)$-RDP in one training iteration of DPVFLGAN-TS.*

Proof. Let $X_i \in \mathbb{R}^{N\times|A_i|\times T}$ denote the local data of party i and $\boldsymbol{x}_i^B$ and $\boldsymbol{x}_i^{B'} \in \mathbb{R}^{B\times|A_i|\times T}$ denote two adjacent mini-batches sampled from X_i, where N is dataset size, B is mini-batch size, A_i is attribute size of party i, and T is length of the time series. Then, $\boldsymbol{x}_{\boldsymbol{ij}}^B \in \mathbb{R}^{B\times T}$ is the attribute that is input into D_{ij} and $\boldsymbol{x}_{\boldsymbol{i}}^B$ is the input of FE_i.

The gradients of the first-layer parameters of $M \in \{D_{i1}, \cdots, D_{i|A_i|}, FE_i\}$ are clipped using (16). Then, the L2 norm of those gradients has the following upper bound,

$$\left\| clip(\mathcal{G}^1_{M(\boldsymbol{x}_j^B)}, C) \right\|_2 \leq C. \tag{39}$$

According to the triangle inequality, the L2 sensitivity of the parameters can be derived as

$$\Delta_2 f = \max_{\boldsymbol{x}_i^B, \boldsymbol{x}_i^{B'}} \left\| clip(\mathcal{G}^1_{M(\boldsymbol{x}_i^B)}, C) - clip(\mathcal{G}^1_{M(\boldsymbol{x}_i^{B'})}, C) \right\|_2 \leq 2C. \tag{40}$$

According to Proposition 4, $\mathcal{G}_M^1$ computed by (17) satisfies $(\alpha, \alpha/(2\sigma^2))$-RDP.

According to Proposition 5, the parameters of the first layer of M updated by,

$$\boldsymbol{\theta}_M^1 = \boldsymbol{\theta}_M^1 - \eta_M \mathcal{G}_M^1, \tag{41}$$

satisfy the same RDP as $\mathcal{G}_M^1$. The function of M, f_M, can be expressed as,

$$f_M = func_M(\boldsymbol{\theta}_M^1 \boldsymbol{x}_{i(j)}), \tag{42}$$

where $\boldsymbol{x}_{i(j)}$ denotes the input of M, $func_M$ denotes the calculation after the first layer, i.e., $\boldsymbol{\theta}_M^1 \boldsymbol{x}_i(j)$. Thus, according to Proposition 5, since $\boldsymbol{\theta}_{D_i}^1$ satisfies $(\alpha, \alpha/(2\sigma^2))$-RDP, the mechanism f_M in (42) satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP, i.e., all the local attribute discriminators and feature extractors satisfy $(\alpha, \alpha/(2\sigma^2))$-RDP.

On the other hand, the local attribute generator, G_{ij}, is trained by the corresponding discriminator D_{ij} and feature extractor FE_i. During the back-propagation process, let $\boldsymbol{\delta}_{FE_i}^1$ and $\boldsymbol{\delta}_{D_i}^1$ denote the backward gradients after the first layer of FE_i and D_i, respectively. Then, the backward gradients for G_{ij}, $\boldsymbol{\delta}^{G_{ij}}$, can be calculated by,

$$\boldsymbol{\delta}^{G_{ij}} = \boldsymbol{\delta}_{D_i}^1 \boldsymbol{\theta}_{D_i}^1 + \boldsymbol{\delta}_{FE_i}^1 \boldsymbol{\theta}_{FE_i}^1. \tag{43}$$

According to Proposition 1 (in the main paper), $\boldsymbol{\delta}^{G_{ij}}$ satisfies $(\alpha, \alpha/\sigma^2)$-RDP. Since the parameters of G_{ij} is updated according to $\boldsymbol{\delta}^{G_{ij}}$, G_{ij} satisfies $(\alpha, \alpha/\sigma^2)$-RDP. □

C Threat Model of the Auditing Scheme

We consider the scenario where all models, including attribute generators, attribute discriminators, feature extractors, and the shared discriminator, are kept private while the generated synthetic dataset is publicly accessible; that is, the attacker can only access the synthetic dataset.

In [41], the authors proposed a shadow model-based membership inference attack that assumes the adversary has an auxiliary dataset with a similar distribution to the private datasets and is much larger than the private dataset. However, this assumption is impractical for data owners since they intend to use most data to train the model rather than audit privacy breaches. Thus, this paper uses the Leave-One-Out (LOO) assumption proposed in [58], which is much stronger than the auxiliary data assumption. In LOO assumption, the attacker knows the whole training dataset but **one** target sample and aims to guess the existence of the target sample.

Takeaway: LOO is a strong assumption. The privacy breaches of the synthetic dataset will be negligible under realistic assumptions if the attack success rate is low under the LOO assumption.

D Construction of Six-Feature Sine Dataset

Synthetic Sine Datasets: Similar to the two-attribute Sine dataset, the six-attribute dataset is constructed as follows,

$$F_{1i} = Asin(2\pi f_{1i}\boldsymbol{t}) + \epsilon, \tag{44}$$

$$F_{2i} = Asin(2\pi f_{2i}\boldsymbol{t}) + \epsilon, \quad i \in \{1, 2, 3\}, \tag{45}$$

where F_{ji} denotes the attribute i stored in Party j, A can be sampled from $\mathcal{N}(0.4, 0.05)$ or $\mathcal{N}(0.6, 0.05)$ with equal propability, $\epsilon \in \mathcal{N}(0, 0.05)$, and $[f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}] = [0.01, 0.005, 0.0075, 0.0125, 0.015, 0.0175]$. In summary, in the two-party scenario, each party stores three attributes, and the amplitudes of the six attributes are the same for a given sample. We can utilize this characteristic to evaluate whether the generative methods learn the correlation between the six attributes stored in different parties. We generate 1,024 samples for both class, i.e., $A \in \mathcal{N}(0.4, 0.05)$ and $A \in \mathcal{N}(0.6, 0.05)$, and each attribute consists of 800 time steps ($\boldsymbol{t} = [0, 1, \cdots, 799]$ in (44) and (45)).

Received 25 February 2025; revised 19 September 2025; accepted 5 November 2025