Federated Time Series Generation on Feature and Temporally Misaligned Data

Zhi Wen Soi[1], Chenrui Fan[1], Aditya Shankar[2], Abel Malan[3], and Lydia Y. Chen[2,3(✉)]

[1] University of Bern, Bern, Switzerland
{zhi.soi,chenrui.fan}@students.unibe.ch
[2] TU Delft, Delft, The Netherlands
a.shankar@tudelft.nl, lydiachen@ieee.org
[3] University of Neuchâtel, Neuchâtel, Switzerland
abele.malan@unine.ch

Abstract. Distributed time series data presents a challenge for federated learning, as clients often possess different feature sets and have misaligned time steps. Existing federated time series models are limited by the assumption of perfect temporal or feature alignment across clients. In this paper, we propose FedTDD, a novel federated time series diffusion model that jointly learns a synthesizer across clients. At the core of FedTDD is a novel data distillation and aggregation framework that reconciles the differences between clients by imputing the misaligned timesteps and features. In contrast to traditional federated learning, FedTDD learns the correlation across clients' time series through the exchange of local synthetic outputs instead of model parameters. A coordinator iteratively improves a global distiller network by leveraging shared knowledge from clients through the exchange of synthetic data. As the distiller becomes more refined over time, it subsequently enhances the quality of the clients' local feature estimates, allowing each client to then improve its local imputations for missing data using the latest, more accurate distiller. Experimental results on five datasets demonstrate FedTDD's effectiveness compared to centralized training, and the effectiveness of sharing synthetic outputs to transfer knowledge of local time series. Notably, FedTDD achieves 79.4% and 62.8% improvement over local training in Context-FID and Correlational scores. Our code is available at: https://github.com/soizhiwen/FedTDD.

Keywords: Federated Learning · Generative Models · Time Series

C. Fan and A. Shankar—Equal contribution.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-032-05981-9_23.

R. P. Ribeiro et al. (Eds.): ECML PKDD 2025, LNAI 16014, pp. 384–399, 2026.
https://doi.org/10.1007/978-3-032-05981-9_23

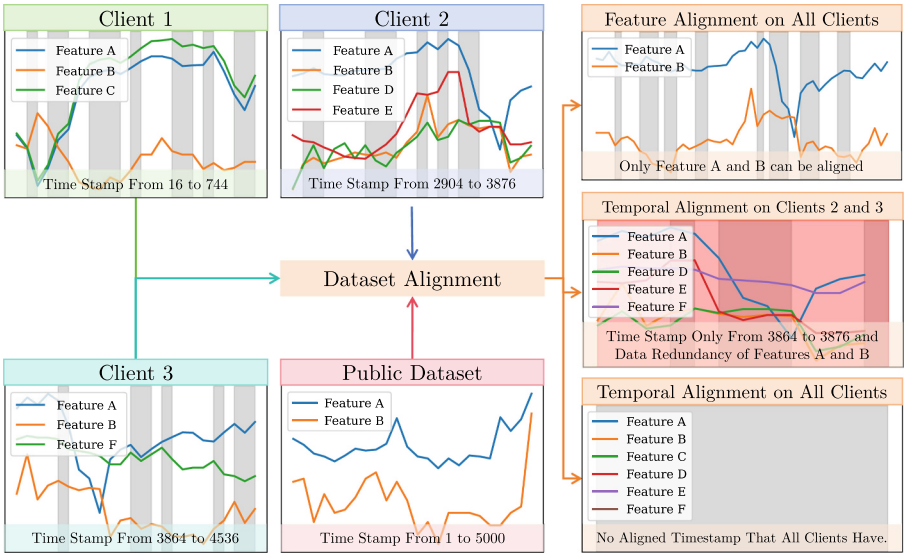


Fig. 1. Feature and temporally misaligned time series. The grey masking indicates missing data. (Color figure online)

1 Introduction

Multivariate time series data are pivotal in many domains, such as healthcare, finance, manufacturing, and sales [16]. Consider a collaboration between multiple clients, shown in Fig. 1. In a healthcare setting, these clients could be hospitals, each collecting patient data locally for a downstream task, such as predicting patient outcomes. The data gathered, such as vital signs like heart rate and blood pressure, is inherently *temporal*, i.e., time series data. Aggregating data from all the sources could improve model performance due to increased sampled diversity when training downstream predictive models. However, privacy regulations such as the General Data Protection Regulation (GDPR) and confidentiality agreements between hospitals prevent sharing of raw data [1,20,32].

Federated learning (FL) [19] takes a step towards tackling this privacy challenge by enabling clients to train a global model by sharing locally trained model parameters rather than raw data. However, this environment faces the challenge of *feature and temporal misalignment* [18], as hospitals may possess different feature sets with varying time intervals for data collection.

In *horizontal* FL [14], different clients have data for the same features but for different samples or timesteps. Hence, it can tackle situations involving temporal misalignment but not feature misalignment. On the other hand, in *vertical* FL [17], different clients possess different feature sets for the same samples or timesteps. While this can handle feature misalignment, it cannot tackle temporal

misalignment. Hence, neither horizontal nor vertical FL can fully tackle scenarios with both feature and temporal misalignment. On top of this, data may be missing or incomplete due to unavailability or inconsistent collection frequencies, further hindering a model's ability to learn patterns [23].

To overcome these limitations, we propose **FedTDD** (Federated Learning in Multivariate Time Series via Data Distillation), a first-of-its-kind federated time series diffusion model capable of learning a time series synthesizer from clients' distinct features with temporal misalignment. FedTDD introduces a novel data distillation [27] and aggregation framework for the common feature set, whose values differ across clients and can be obtained from the public domain. In this framework, a coordinator maintains a global model called the *distiller*, trained iteratively using a combination of public data and clients' intermediate synthetic data outputs. Each client keeps a local time series diffusion model for imputing local features which leverages the latest distiller to improve the quality of local estimates. Unlike traditional federated learning, FedTDD learns the correlations among clients' time series through the exchange of synthetic outputs instead of aggregating models [19], effectively handling feature and temporal misalignment without sharing raw data.

Given the recent advancements of diffusion models over mainstream generative models like *Generative Adversarial Networks* (GANs) [7], we utilize a time series *Denoising Diffusion Probabilistic Model* (DDPM) [9], adapted to handle temporal dependencies through temporal embeddings and sequential conditioning. Specifically, we select **Diffusion-TS** [35] since it leverages both time and frequency domain information, effectively capturing trends and seasonality, which leads to a more accurate imputation of missing data. By imputing data from unaligned time steps, clients can obtain temporally aligned data without needing alignment on the features or sharing raw data.

In summary, our major contributions are as follows: (i) We propose a novel federated generative learning framework that effectively handles temporal and feature-level misalignment and data missing problems in time series data. (ii) We develop a data distillation and aggregation framework that learns correlations among clients' time series by exchanging synthetic data instead of model parameters, enabling clients to improve their local models without direct data sharing and effectively handling data discrepancies. (iii) We conduct extensive experiments on five benchmark datasets, showing up to 79.4% and 62.8% improvement over local training in Context-FID and Correlational scores under extreme feature and temporal misalignment cases and achieving performance comparable to centralized training.

2 Related Work

Time Series Generation. Generative models for time series data aim to capture temporal dependencies and sequential patterns inherent in such datasets. TimeGAN [34] combines *generative adversarial networks* (GANs) [7] with

Table 1. Overview of the related work.

Method	Model Type	Time Series	FL Type	Handles Temporal Misalignment	Handles Feature Misalignment
GTV [38]	GAN	×	Vertical	×	✓
DPGDAN [33]	GAN	×	Vertical	×	✓
SiloFuse [28]	DDPM	×	Vertical	×	✓
VFLGAN-TS [36]	GAN	✓	Vertical	×	✓
FedGAN [25]	GAN	✓	Horizontal	✓	×
T2TGAN [3]	GAN	✓	Horizontal	✓	×
FedTDD (Ours)	DDPM	✓	Hybrid	✓	✓

recurrent neural networks [22] to produce realistic multivariate time series. TimeVAE [5] utilizes variational autoencoders (VAEs) [11] tailored for time series to capture trends and seasonality. Recently, diffusion-based models like TimeGrad [26], CSDI [30], SSSD [2], TSDiff [13], and Diffusion-TS [35] have further advanced time series generation by producing high-fidelity sequences, outperforming the mainstream GANs and VAE-based techniques. Despite their effectiveness, these models operate in centralized settings and assume fully aligned data with consistent features and timestamps. They are not equipped to handle feature and temporal misalignments common in real-world distributed scenarios, making them unsuitable for federated environments with heterogeneous data distributions [21,24].

Federated Learning with Generative Models. Federated learning [37] has primarily been applied to image generation, such as FedCycleGAN [29] leverages CycleGAN [39] in federated settings to generate synthetic images while preserving data privacy. For tabular data, methods like GTV [38], DPGDAN [33], and SiloFuse [28] employ GANs and diffusion models within vertical federated learning frameworks to synthesize tabular datasets. However, these approaches focus on vertically partitioned data, where all clients have features corresponding to the same sample ID, and do not address data redundancy or misalignment issues. Federated learning with generative models for time series data remains under-explored. Existing works such as FedGAN [25], VFLGAN-TS [36], and T2TGAN [3] extend GANs to federated time series generation. VFLGAN-TS operates in a vertical federated learning context, tackling feature misalignment, but does not handle temporal misalignment. In contrast, T2TGAN tackles horizontal federated learning settings but introduces data redundancy due to overlapping data among clients and cannot handle feature mismatches between clients. As summarized in Table 1, these methods encounter issues as shown in Fig. 1, making them less effective for federated time series generation where both feature and temporal misalignments are prevalent.

Preliminary on Generative Modeling with DDPMs. For the generative backbone, we adopt the Diffusion-TS architecture [35], which extends DDPMs [9] to capture temporal patterns using a generative modeling process. DDPMs are models trained using a *forward noising* and *backward denoising* process. The forward phase progressively adds random Gaussian noise to the data $\mathbf{s}_0$ at diffusion step t,

where the transition is parameterized by $q(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = \mathcal{N}(\mathbf{s}_t; \sqrt{1-\beta_t}\,\mathbf{s}_{t-1}, \beta_t\,\mathbf{I})$ with $\beta_t \in (0,1)$, eventually transforming it into pure noise $\mathbf{s}_T \sim \mathcal{N}(0, \mathbf{I})$. The backward phase is where the model learns to reverse this noising process. Starting from random noise $\mathbf{s}_T \sim \mathcal{N}(0, \mathbf{I})$, it iteratively removes the added noise step by step via $p_\theta(\mathbf{s}_{t-1} \mid \mathbf{s}_t) = \mathcal{N}(\mathbf{s}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{s}_t, t), \Sigma_\theta(\mathbf{s}_t, t))$, to reconstruct a new data sample resembling the original input distribution. The functions $\boldsymbol{\mu}_\theta$ and Σ_θ are generally estimated using a model.

Diffusion-TS extends standard DDPMs by incorporating mechanisms specifically designed for time series characteristics such as trends and seasonality [12]. Instead of treating data points independently, it utilizes an encoder-decoder transformer architecture [31] that processes entire sequences, effectively modeling temporal relationships. To handle trends, Diffusion-TS decomposes the time series into components that represent slow-varying behaviors over time. For capturing seasonality and periodic patterns, it employs frequency domain analysis using the Fast Fourier Transform (FFT) [8]. By integrating FFT, the model can analyze and reconstruct cyclical patterns [4] within the data, allowing it to learn both time and frequency domain representations [6]. This combination enables Diffusion-TS to generate more accurate and realistic time series data by effectively modeling complex temporal dynamics. Besides, Diffusion-TS supports both *unconditional* and *conditional* generation. In the unconditional generation, the model produces new samples solely based on the learned data distribution, starting from random noise and applying the learned denoising process. In the conditional generation, Diffusion-TS utilizes gradient-based guidance during sampling to incorporate the observed data $\mathbf{y}$. At each diffusion step, the model refines its estimated time series $\hat{\mathbf{s}}_0$ by adjusting it with a gradient term that enforces consistency with the observed data. The refinement can be computed via $\tilde{\mathbf{s}}_0(\mathbf{s}_t, t; \theta) = \hat{\mathbf{s}}_0(\mathbf{s}_t, t; \theta) + \eta \nabla_{\mathbf{s}_t}(\|\mathbf{y} - \hat{\mathbf{s}}_0(\mathbf{s}_t, t; \theta)\|^2 + \gamma \log p(\mathbf{s}_{t-1} \mid \mathbf{s}_t))$, where η is a hyperparameter that controls the strength of the gradient guidance, and γ balances the trade-off between fitting the observed data and maintaining the generative model's prior distribution $p(\mathbf{s}_{t-1} \mid \mathbf{s}_t)$. This iterative refinement ensures that the generated time series aligns with the provided observations and preserves the temporal patterns learned during training. Further details of Diffusion-TS are shown in Appendix B.2.

3 FedTDD

In this work, we address the problem of collaborative time series imputation in the presence of temporal and feature misalignments, without requiring the sharing of raw data. In a federated learning setting, clients may possess different subsets of features. We categorize features into two types: *common features* and *exclusive features*. Common features are those present in all clients and also available in a public dataset, while exclusive features are unique to each client and not shared. For example, market indices might be common features in financial data, while individual portfolio holdings are exclusive. Our proposed framework, FedTDD, as shown in Fig. 2, tackles this problem using two models. A global

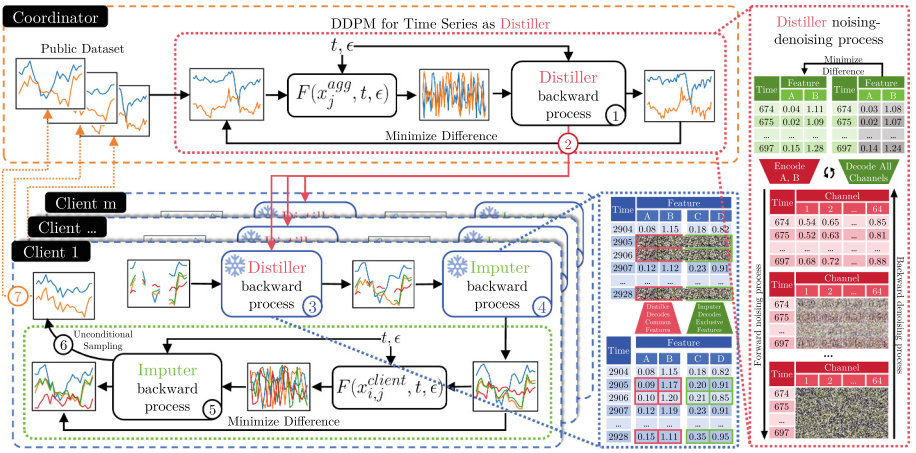


Fig. 2. FedTDD Structure. First, the Distiller is pre-trained on a public dataset. Then, each client uses the distiller and imputer to impute common and exclusive features, respectively. Finally, synthetic data is sent back to the coordinator to expand the public dataset for the next round. The order of execution(1-7) is labeled in the figure. Here the common features are A and B, and the exclusive features are C and D.

distiller first imputes missing common features across clients. Local imputer models then use the imputed common features to predict the missing exclusive features for each client, addressing both temporal and feature misalignments. Furthermore, clients protect their privacy by sharing only synthetic versions of the common features while collaboratively improving the global distiller. This cycle of iterative imputation and model refinement ultimately converges to yield good quality imputations, while ensuring that no raw data is shared.

3.1 Problem Definition

We consider a federated learning setup involving N clients and a coordinator. Each client i possesses a time series dataset, denoted as $\mathbf{X}^i = \left[X^i_{j,k}\right]_{\{j=1\ldots T^i, k=1\ldots C^i\}}$, where T^i is the number of time steps, and C^i is the number of channels. These datasets can be split into two components, one for the common features and one for the exclusive features, i.e., $\mathbf{X}^i = \mathbf{X}^i_{\text{comm}} \cup \mathbf{X}^i_{\text{ex}}$. The coordinator holds a public dataset $\mathbf{X}^{\text{pub}} = \left[X^{\text{pub}}_{j,k}\right]_{\forall j; k \in \mathcal{F}_{\text{comm}}}$, which contains data for the common features $\mathcal{F}_{\text{comm}}$ but without any missing values. This public dataset is time-indexed differently from the clients' data and provides a reliable reference for the common features. Each client's time series data comes from a distinct time interval, meaning that each client's time indices j are unique. The feature set for each client i, $\mathcal{F}^i$, consists of common features $\mathcal{F}_{\text{comm}}$, which are shared across all clients, and exclusive features $\mathcal{F}^i_{\text{ex}}$, which are specific to each client. Thus, the overall feature set for client i is represented as

$\mathcal{F}^i = \mathcal{F}_{\text{comm}} \cup \mathcal{F}^i_{\text{ex}}$. Conversely, clients may have missing values in both the common and exclusive features. These missing values are indicated by a binary mask matrix $\mathbf{M}^i = \left[M^i_{j,k}\right]_{\forall j,k}$, where $M^i_{j,k} = 1$ if the value $X^i_{j,k}$ is observed while 0 indicates it is missing. The mask can be split into two parts: $\mathbf{M}^i_{\text{comm}}$, which corresponds to missing data in the common features, and $\mathbf{M}^i_{\text{ex}}$, which corresponds to missing data in the exclusive features. The goal is to design a collaborative method that enables clients to leverage shared knowledge and the public dataset to input the missing data locally without sharing raw data. Appendix A summarizes the mathematical notations used.

3.2 Hybrid Federated Learning for Imputation Under Misalignment

Algorithm 1 presents the overview of FedTDD. The framework consists of two key components: the global distiller model $\mathcal{D}$ and the local imputer models $\mathcal{U}^i$. The global distiller $\mathcal{D}$ imputes missing common features shared across all clients, while each client trains a local imputer $\mathcal{U}^i$ to infer missing exclusive features specific to their data. These components work together to address temporal and feature misalignment by iteratively improving the imputation process over several rounds r ranging from 1 to R.

Algorithm 1: FedTDD

Input: Public dataset $\mathbf{X}^{\text{pub}}$, clients' datasets $\mathbf{X}^i$
Result: Global distiller model $\mathcal{D}$, local imputer models $\mathcal{U}^i$

Initialize: Train $\mathcal{D}$ on $\mathbf{X}^{\text{pub}}$
for $r = 1$ *to* R **do**
  for *each client* i **do**
    Receive global distiller $\mathcal{D}$
    $\hat{\mathbf{X}}^i_{\text{comm}} \leftarrow \mathcal{D}\left(\mathbf{X}^i_{\text{comm}}, \mathbf{M}^i_{\text{comm}}\right)$; ▷ Impute common features
    $\hat{\mathbf{X}}^i_{\text{ex}} \leftarrow \mathcal{U}\left(\mathbf{X}^i_{\text{ex}}, \mathbf{M}^i_{\text{ex}}\right)$; ▷ Impute exclusive features
    $\mathbf{X}^i_{\text{train}} \leftarrow \hat{\mathbf{X}}^i_{\text{comm}} \cup \hat{\mathbf{X}}^i_{\text{ex}}$; ▷ Combine with exclusive features
    Train $\mathcal{U}^i$ on $\mathbf{X}^i_{\text{train}}$; ▷ Train local imputer
    $\hat{\mathbf{X}}^i \leftarrow \mathcal{U}^i(\mathbf{z}), \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$; ▷ Generate synthetic data
    Send $\hat{\mathbf{X}}^i_{\text{comm}}$ from $\hat{\mathbf{X}}^i$ to coordinator
  end
  for *each client* i **do**
    Select $n_r = \frac{r}{R}\,\alpha \cdot L$ sequences from $\hat{\mathbf{X}}^i_{\text{comm}}$
    $\mathbf{X}^{\text{pub}} \leftarrow \mathbf{X}^{\text{pub}} \cup \hat{\mathbf{X}}^i_{\text{comm}}[1 : n_r]$; ▷ Expand public dataset
  end
  Finetune $\mathcal{D}$ on updated $\mathbf{X}^{\text{pub}}$
end

The process begins with the coordinator training a global distiller model $\mathcal{D}$ using the public dataset $\mathbf{X}^{\text{pub}}$. $\mathcal{D}$ leverages a temporal DDPM backbone to apply

a forward diffusion process by gradually adding noise to the data and learns to reverse this process. During training, $\mathcal{D}$ conducts *unconditional generation* by starting from Gaussian noise $\boldsymbol{\epsilon}$ and learning to approximate the data distribution through the time and frequency domain components [35]. Formally, we have

$$\mathcal{L}_{\text{time}} = \mathbb{E}_{(j,k,t)\,|\,\mathbf{M}^{\text{pub}}_{j,k}=1}\left[\left\|\mathbf{X}^{\text{pub}}_{j,k} - \tilde{\mathbf{X}}^{\text{pub}}_{j,k}(\mathbf{X}^{\text{pub}}_{j,k,t}, t, \boldsymbol{\epsilon}; \theta)\right\|^2\right] \quad \text{and} \tag{1}$$

$$\mathcal{L}_{\text{freq}} = \mathbb{E}_{(j,k,t)\,|\,\mathbf{M}^{\text{pub}}_{j,k}=1}\left[\left\|\text{FFT}(\mathbf{X}^{\text{pub}}_{j,k}) - \text{FFT}\left(\tilde{\mathbf{X}}^{\text{pub}}_{j,k}(\mathbf{X}^{\text{pub}}_{j,k,t}, t, \boldsymbol{\epsilon}; \theta)\right)\right\|^2\right], \tag{2}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, $\mathbf{X}^{\text{pub}}_{j,k}$ is the (j,k)-th entry of $\mathbf{X}^{\text{pub}}$, $\tilde{\mathbf{X}}^{\text{pub}}_{j,k}$ is the denoised estimate from $\mathcal{D}$, and FFT denotes the Fast Fourier Transform [8], which is a mathematical operation that converts a finite-length time domain signal to its frequency domain representation. We take the following objective

$$\mathcal{L}_{\text{distiller}(\mathcal{D}^i)} = \mathbb{E}_{(j,k,t)\,|\,\mathbf{M}^{\text{pub}}_{j,k}=1}\left[w_t\left(\lambda_1\mathcal{L}_{\text{time}} + \lambda_2\mathcal{L}_{\text{freq}}\right)\right], \quad w_t = \frac{\lambda\gamma_t(1-\bar{\gamma}_t)}{\delta_t^2}, \tag{3}$$

where λ_1 and λ_2 control the balance between time and frequency losses while w_t emphasizes learning at larger diffusion steps, with λ being a small constant. The parameter $\delta_t \in (0,1)$ determines the amount of noise added at each forward diffusion step, where t is a diffusion time step uniformly sampled from 1 to T during training. The cumulative product $\bar{\gamma}_t = \prod_{v=1}^{t}\gamma_v$, with $\gamma_t = 1 - \delta_t$, track how the original signal diminishes over time due to the added noise. By weighting the loss at different steps, w_t helps the model focus on reconstructing the signal under high-noise conditions.

After this initial training, the coordinator distributes the trained global distiller model $\mathcal{D}$ to all participating clients. Each client i then utilizes $\mathcal{D}$ to impute their missing common features. Since clients may have missing values in $\mathbf{X}^i_{\text{comm}}$, they input their data along with the corresponding mask $\mathbf{M}^i_{\text{comm}}$ to the distiller model, which will perform *conditional generation* to iteratively refine the imputed data by sampling from the conditional distribution guided by the observed data, shown in Eq. 17 in Appendix B.2. The imputation process follows $\hat{\mathbf{X}}^i_{\text{comm}} = \mathcal{D}(\mathbf{X}^i_{\text{comm}}, \mathbf{M}^i_{\text{comm}})$, where $\mathcal{D}$ reconstructs only the missing values, indicated by $\mathbf{M}^i_{\text{comm}} = 0$. Similarly, the local imputer imputes missing values in $\mathbf{X}^i_{\text{ex}}$ by inputting their data along with the corresponding mask $\mathbf{M}^i_{\text{ex}}$ to the imputer model via $\hat{\mathbf{X}}^i_{\text{ex}} = \mathcal{U}(\mathbf{X}^i_{\text{ex}}, \mathbf{M}^i_{\text{ex}})$. The imputed common features $\hat{\mathbf{X}}^i_{\text{comm}}$ are then combined with the available exclusive features $\hat{\mathbf{X}}^i_{\text{ex}}$ to form the training data $\mathbf{X}^i_{\text{train}} = \hat{\mathbf{X}}^i_{\text{comm}} \cup \hat{\mathbf{X}}^i_{\text{ex}}$ for the local imputer. Meanwhile, each client trains their local imputer model $\mathcal{U}^i$ using $\mathbf{X}^i_{\text{train}}$ as the ground truth. Since the imputed common features $\hat{\mathbf{X}}^i_{\text{comm}}$ are fully known (as they are outputs from the pre-trained and fine-tuned $\mathcal{D}$), they are entirely used as ground truth for training $\mathcal{U}^i$, regardless of the original mask $\mathbf{M}^i_{\text{comm}}$. For the exclusive features, only the observed entries indicated by the mask $\mathbf{M}^i_{\text{ex}}$ are used as ground truth since the quality of the imputer's generated data during training is not sufficient to be used as ground truth. We define the loss mask as $\mathbf{M}^i_{\text{loss}} = \mathbf{1}^i_{\text{comm}} \cup \mathbf{M}^i_{\text{ex}}$, where

$\mathbf{1}^i_{\text{comm}}$ is a matrix of ones corresponding to the common features of client i. This loss mask ensures that the reconstruction loss is computed over all entries of the imputed common features and the observed entries of the exclusive features. The training loss for the imputer $\mathcal{U}^i$ can be defined as follows:

$$\mathcal{L}_{\text{imputer}(\mathcal{U}^i)} = \mathbb{E}_{(j,k,t)\,|\,\mathbf{M}^i_{\text{loss}_{j,k}}=1}\left[w_t\left(\lambda_1\mathcal{L}^i_{\text{time}} + \lambda_2\mathcal{L}^i_{\text{freq}}\right)\right], \tag{4}$$

$$\text{where}\quad \mathcal{L}^i_{\text{time}} = \mathbb{E}_{(j,k,t)\,|\,\mathbf{M}^i_{\text{loss}_{j,k}}=1}\left[\left\|\mathbf{X}^i_{\text{train}_{j,k}} - \tilde{\mathbf{X}}^i_{\text{train}_{j,k}}(\mathbf{X}^i_{\text{train}_{j,k,t}}, t; \theta)\right\|^2\right]$$

$$\text{and}\quad \mathcal{L}^i_{\text{freq}} = \mathbb{E}_{(j,k,t)\,|\,\mathbf{M}^i_{\text{loss}_{j,k}}=1}\left[\left\|\text{FFT}(\mathbf{X}^i_{\text{train}_{j,k}}) - \text{FFT}\left(\tilde{\mathbf{X}}^i_{\text{train}_{j,k}}(\mathbf{X}^i_{\text{train}_{j,k,t}}, t; \theta)\right)\right\|^2\right],$$

where $\mathbf{X}^i_{\text{train}_{j,k}}$ is the (j,k)-th entry of $\mathbf{X}^i_{\text{train}}$, $\tilde{\mathbf{X}}^i_{\text{train}_{j,k}}$ is the denoised estimate from $\mathcal{U}$. After training, each client uses the trained imputer $\mathcal{U}^i$ to generate a synthetic dataset through *unconditional synthesis*, which includes both the common features $\hat{\mathbf{X}}^i_{\text{comm}}$ and the exclusive features $\hat{\mathbf{X}}^i_{\text{ex}}$. Starting from Gaussian noise, the imputer generates samples $\hat{\mathbf{X}}^i = \mathcal{U}^i(\mathbf{z}), \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, that capture the distribution of both common and exclusive features.

To protect privacy, only the common features from the synthetic dataset, $\hat{\mathbf{X}}^i_{\text{comm}}$, are shared with the coordinator. This ensures that no raw or exclusive client data is exposed during the collaborative learning. The coordinator uses the synthetic common feature data from the clients to expand its public dataset. Rather than simply absorbing all the synthetic data, the coordinator carefully controls the growth of the dataset by accepting a fraction of the sequences from each client. Specifically, the coordinator adds $\frac{r}{R}\alpha * L$, where L represents the length of the synthetic datasets $\hat{\mathbf{X}}^i_{\text{comm}}; \forall i \in \{1, 2, \ldots, N\}$, α is a hyperparameter between 0 and 1, and the ratio of r and R yields a number that linearly increases up to 1, allowing for a gradual expansion as the rounds increase. The coordinator retrains the global distiller $\mathcal{D}$ using this expanded dataset. The addition of synthetic data enhances the distiller's ability to learn the patterns necessary for imputing missing common features.

The overall process creates an iterative cycle of improvement. As clients' generative models, specifically their local imputers, become more accurate with each round, the quality of the synthetic data they generate also improves. This higher-quality synthetic data, in turn, improves the distiller model at the coordinator, which benefits all clients when it is redistributed. Over several training rounds, this mutual reinforcement drives both the global distiller and the local imputers to improve continuously. Ultimately, the process converges, yielding robust imputation models without requiring clients to share their raw data.

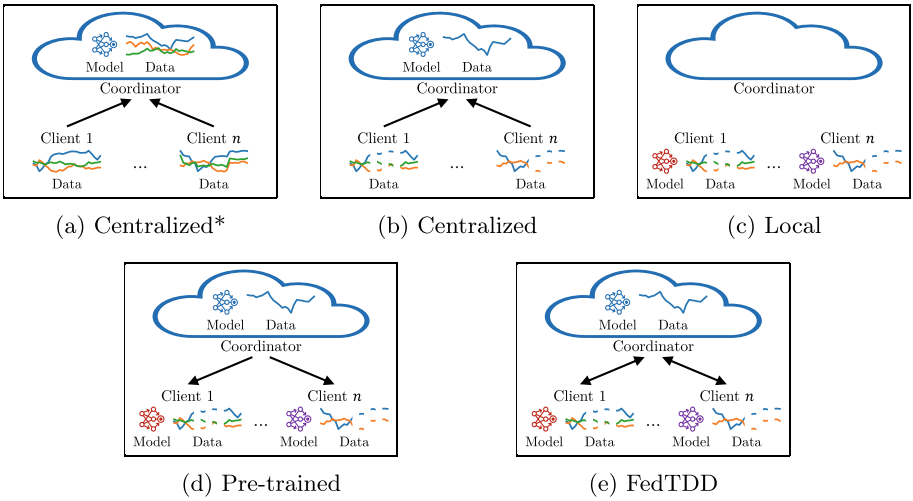
4 Experiments

We assess FedTDD's performance by showing its advantages and disadvantages when applied to multiple benchmark datasets. We leave the analysis of different training configurations in the Appendix C, where we examine the impact of limited public data, abundant sequences with missing data, imbalanced data distributions and different aggregation strategies on model performance.

Datasets. To assess the quality of synthetic data, we consider four real-world datasets and one simulated dataset with different properties, such as the number of features, correlation, periodicity, and noise levels. Each dataset is preprocessed using a sliding window technique [34] to segment the data into sequences of length 24 to capture meaningful temporal dependencies while keeping the computational cost manageable. **Stocks** is the daily historical Google stock data from 2004 to 2019 with highly correlated features. **ETTh** recorded the electricity transformers hourly between July 2016 and July 2018, including load and oil temperature data that consists of 7 features. **Energy** from UCI appliances energy prediction dataset with 10-minute intervals for about 4.5 months. **fMRI** is a realistic simulation of brain activity time series with 50 features. **MuJoCo** is a physics-based simulation time series containing 14 features. We show the statistics of all datasets in Appendix D.2.

Baselines. We compare FedTDD against approaches show in Fig. 3, 3b, 3c and 3d. For the **Centralized*** training, we aggregate all data from individual clients, including public data, into a single location, where a global model is trained using the combined dataset, and this will be trained with all available features in the dataset and without missing values. While **Centralized** uses the same training procedure as Centralized*, it is, however, trained on a combined dataset with missing values and corresponding features available from each client plus the public data. To deal with differing features across clients, we create the combined dataset consisting of the total number of features in the particular benchmark dataset and zero-fill any remaining features to ensure uniformity. On the other hand, **Local** training involves training a separate model for each client using only their local data, without any communication or data aggregation. This approach has to be done to verify that FedTDD can perform relatively better than train locally. Finally, the **Pre-trained** approach leverages a model trained on a public dataset and uses it to impute the common features in local data from each client. Again, there is no data aggregation for this approach. In comparison, FedTDD integrates the Pre-trained approach and applies data aggregation to it. We utilized a SOTA diffusion-based multivariate time series generative model, Diffusion-TS [35], as the backbone for these baselines and FedTDD. Alternatively, any other time series generative model can be adopted in these approaches in a plug-and-play manner.

Evaluation Metrics. We quantitatively assess the quality of the generated synthetic data using four key metrics (see Appendix D.3 for more details). **Context-Fréchet Inception Distance (Context-FID) score** [10] evaluates the similarity between the distribution of real and synthetic time series data by computing the Fréchet distance. **Correlational score** [15] measures the correlation between the features of multivariate time series in the synthetic data compared to its real data. **Discriminative score** [34] measures the realism of the synthetic data by training a binary classifier to distinguish between real and synthetic data. **Predictive score** [34] evaluates the utility of the synthetic data by training a sequence-to-sequence model on the synthetic data and measuring



(a) Centralized* (b) Centralized (c) Local

(d) Pre-trained (e) FedTDD

Fig. 3. Illustrations of different baselines compared to FedTDD. The data in the coordinator, also called public data, in Fig. 3b, 3d and 3e consists only common features time series. Dashes indicate temporal missing values.

Table 2. Results on multiple time series datasets. **Bold** indicates best performance.

Metric	Method	Stocks	ETTh	MuJoCo	Energy	fMRI
Context-FID	Centralized*	0.682 ± 0.106	0.281 ± 0.040	0.782 ± 0.138	0.533 ± 0.082	1.737 ± 0.125
	Centralized	3.548 ± 0.990	8.870 ± 2.295	10.00 ± 2.814	9.343 ± 2.808	13.56 ± 3.357
	Local	1.648 ± 0.229	1.313 ± 0.188	0.751 ± 0.121	1.179 ± 0.179	1.694 ± 0.153
	Pre-trained	1.047 ± 0.169	0.326 ± 0.040	0.617 ± 0.090	0.412 ± 0.054	**1.411 ± 0.102**
	FedTDD	**0.675 ± 0.087**	**0.271 ± 0.038**	**0.529 ± 0.068**	**0.376 ± 0.056**	1.459 ± 0.099
Correlational	Centralized*	0.061 ± 0.043	0.253 ± 0.094	1.989 ± 0.247	5.231 ± 1.294	7.900 ± 0.384
	Centralized	0.769 ± 0.336	0.340 ± 0.097	2.230 ± 0.518	5.681 ± 0.634	18.07 ± 2.311
	Local	0.156 ± 0.120	0.239 ± 0.079	1.298 ± 0.260	3.447 ± 0.838	**5.992 ± 0.383**
	Pre-trained	0.077 ± 0.052	0.165 ± 0.074	1.323 ± 0.171	2.821 ± 0.651	6.049 ± 0.349
	FedTDD	**0.058 ± 0.050**	**0.161 ± 0.064**	**1.296 ± 0.215**	**2.800 ± 0.686**	6.017 ± 0.364
Discriminative	Centralized*	0.136 ± 0.091	0.199 ± 0.061	0.297 ± 0.108	0.230 ± 0.080	0.422 ± 0.074
	Centralized	0.476 ± 0.042	0.475 ± 0.017	0.474 ± 0.024	0.496 ± 0.006	0.477 ± 0.030
	Local	0.340 ± 0.153	0.298 ± 0.060	0.200 ± 0.092	0.329 ± 0.087	**0.397 ± 0.061**
	Pre-trained	**0.175 ± 0.117**	0.115 ± 0.060	0.208 ± 0.068	**0.141 ± 0.068**	0.419 ± 0.051
	FedTDD	0.185 ± 0.105	**0.106 ± 0.061**	**0.153 ± 0.120**	0.153 ± 0.072	0.414 ± 0.051
Predictive	Centralized*	0.040 ± 0.000	0.127 ± 0.003	0.112 ± 0.015	0.292 ± 0.009	0.137 ± 0.004
	Centralized	0.047 ± 0.012	0.223 ± 0.020	0.165 ± 0.060	0.427 ± 0.053	0.233 ± 0.051
	Local	0.043 ± 0.003	0.118 ± 0.011	0.048 ± 0.006	0.204 ± 0.012	0.135 ± 0.006
	Pre-trained	0.046 ± 0.001	0.104 ± 0.004	0.052 ± 0.004	0.177 ± 0.005	0.133 ± 0.006
	FedTDD	**0.041 ± 0.001**	**0.101 ± 0.004**	**0.048 ± 0.004**	**0.175 ± 0.006**	**0.133 ± 0.004**

its performance on real data. All evaluation metrics are computed based on the respective features of the individual clients and then averaged over five trials, followed by calculating the overall average across the number of clients. The quality of synthetic data is considered the "best" when all metrics approach 0, meaning lower values indicate better quality.

Training Configurations. We run FedTDD and the baselines mentioned above with ten clients, five global rounds, 7500 local epochs for the first round, and 5000 for the rest. Besides, the coordinator trains on the public data consisting of common features, and each client contributes a set of features, which is the combination of common and exclusive features. The number of common features is around 50other hand, we use public ratio (PR) to manipulate the proportion of the public data that has to be reserved from the entire dataset before partitioning the dataset to all clients. Split ratio (SR) divides all sequences into two groups. In the first group, a mask is applied to just the common features, while in the second group, the mask is applied to all features. Moreover, missing ratio (MR) is the missing rate to mask on a sequence of multivariate time series, and we consider the missing scenario as shown in Appendix D.4. In the main experiments, we set PR, SR, and MR to 0.5. All the hyperparameters are listed in Appendix D.5.

4.1 Time Series Generation

In Table 2, we quantitatively analyze the quality of unconditionally generated 24-length time series for diverse time series datasets. FedTDD shows a strong performance comparable to the Centralized* approach. The proposed aggregation mechanism during fine-tuning proved essential to prevent the degradation of the coordinator model's performance and, in turn, the client models. By doing this, we achieved strong results across most datasets. We also present the generated synthetic samples of one representative client for ETTh and fMRI datasets in Fig. 4.

Challenges on fMRI Dataset. We observe that the fMRI dataset's imputation quality was lower than other datasets, as the mean square error between the imputed and real data is greater. Consequently, client models degraded due to training on low-quality imputed data. This suggests that the imputation strategy may need further refinement for such datasets, where the data distribution and complexity present greater challenges for accurate synthetic data generation and imputation. Besides, the Local approach achieves the best Correlational and Discriminative scores for the fMRI dataset. However, we cannot conclude that training locally is the best overall approach for fMRI. As we mentioned, the low performance of FedTDD and Pre-trained is primarily due to the poor quality of the imputed data, which affects training. This shows the advantage of Local training not relying on imputed data, making it seem better suited for the fMRI dataset compared to FedTDD and Pre-trained.

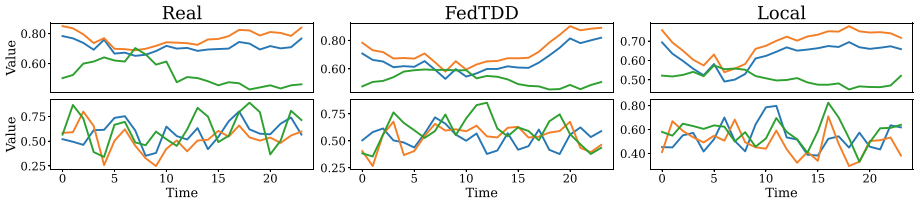


Fig. 4. Real samples and synthetic samples generated unconditionally from FedTDD and Local. The first and second rows of samples are from ETTh and fMRI datasets, respectively.

Table 3. Ablation study for a relatively small number of common features. **Bold** indicates best performance.

Metric	Method	Stocks	ETTh	MuJoCo	Energy	fMRI
Context-FID	Centralized*	0.682 ± 0.106	0.281 ± 0.040	0.782 ± 0.138	0.533 ± 0.082	1.737 ± 0.125
	Centralized	3.733 ± 0.959	11.54 ± 3.894	14.68 ± 4.263	13.17 ± 3.035	15.34 ± 4.789
	Local	1.982 ± 0.234	0.824 ± 0.105	0.660 ± 0.100	0.844 ± 0.127	1.220 ± 0.098
	Pre-trained	0.738 ± 0.142	0.316 ± 0.032	0.547 ± 0.099	0.381 ± 0.066	**1.178 ± 0.104**
	FedTDD	**0.680±0.123**	**0.267±0.036**	**0.510±0.072**	**0.331±0.051**	1.196 ± 0.098
Correlational	Centralized*	0.061 ± 0.043	0.253 ± 0.094	1.989 ± 0.247	5.231 ± 1.294	7.900 ± 0.384
	Centralized	0.697 ± 0.168	0.523 ± 0.095	2.317 ± 0.597	5.781 ± 0.924	31.35 ± 4.923
	Local	0.091 ± 0.052	0.167 ± 0.057	1.079 ± 0.196	1.984 ± 0.594	**4.929±0.395**
	Pre-trained	0.028 ± 0.027	**0.132±0.054**	1.115 ± 0.233	1.795 ± 0.577	5.033 ± 0.323
	FedTDD	**0.025±0.022**	0.137 ± 0.064	**1.060±0.209**	**1.737±0.282**	5.005 ± 0.317
Discriminative	Centralized*	0.136 ± 0.091	0.199 ± 0.061	0.297 ± 0.108	0.230 ± 0.080	0.422 ± 0.074
	Centralized	0.475 ± 0.041	0.469 ± 0.020	0.479 ± 0.026	0.494 ± 0.010	0.484 ± 0.023
	Local	0.300 ± 0.116	0.208 ± 0.070	0.190 ± 0.088	0.241 ± 0.071	**0.398±0.058**
	Pre-trained	0.119 ± 0.088	0.116 ± 0.067	0.163 ± 0.088	0.130 ± 0.058	0.418 ± 0.050
	FedTDD	**0.112±0.097**	**0.107±0.078**	**0.157±0.104**	**0.120±0.067**	0.412 ± 0.057
Predictive	Centralized*	0.040 ± 0.000	0.127 ± 0.003	0.112 ± 0.015	0.292 ± 0.009	0.137 ± 0.004
	Centralized	0.168 ± 0.025	0.196 ± 0.027	0.198 ± 0.049	0.314 ± 0.052	0.223 ± 0.029
	Local	0.084 ± 0.038	0.114 ± 0.009	0.069 ± 0.010	0.199 ± 0.007	**0.130±0.005**
	Pre-trained	0.028 ± 0.007	0.108 ± 0.004	0.063 ± 0.007	0.190 ± 0.005	0.132 ± 0.005
	FedTDD	**0.028 ± 0.005**	**0.107 ± 0.005**	**0.062 ± 0.006**	**0.186 ± 0.004**	**0.130 ± 0.005**

Comparison Between Centralized and Local Training. Both Centralized and Local approaches are trained on datasets with missing values, but their performance differs significantly. This could be due to the different model architectures used in each approach. As aforementioned, the Centralized model is trained on a combined dataset where the additional features are filled with zeros, which results in the worst performance. This shows the advantage of having an individual model trained locally for each client.

4.2 Ablation Study

In Table 3, we show the result of reducing the number of common features in FedTDD. We set the number of common features to around 25% of the total

number of features in the corresponding dataset. As a result, we can observe the robustness of FedTDD when dealing with a relatively small number of common features across most datasets. However, FedTDD does not perform as expected on the fMRI dataset because of the poor quality of imputed data, as mentioned in Sect. 4.1. On the other hand, the performance of Centralized training slightly decreased due to more zeros filling out the combined dataset, especially in the public data.

5 Conclusion

While federated learning is increasingly applied for different regression tasks for time series (TS), it is still limited in handling generative tasks, especially when time series features are vertically partitioned and temporarily misaligned. We propose a novel federated TS generation framework, FedTDD, which trains TS diffusion model by leveraging the self-imputing capability of the diffusion model and globally aggregating from clients' knowledge through data distillation and clients' synthetic data. The central component of FedTDD is a distiller at the coordinator that first is pre-trained on the public datasets and then periodically fine-tuned by the aggregated intermediate synthetic data from the clients. Clients keep their personalized TS diffusion models and train them with local data and synthetic data of the latest distiller periodically. Our extensive evaluation across five datasets shows that FedTDD effectively overcomes the hurdle of feature partition and temporal misalignment, achieving improvements of up to 79.4% and 62.8% over local training on Context-FID and Correlational scores, while delivering performance comparable to centralized baselines.

Acknowledgments. This research is part of the Priv-GSyn, 200021E_229204, of Swiss National Science Foundation and the DEPMAT project, P20-22/N21022, of the research programme Perspectief which is partly financed by the Dutch Research Council (NWO).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alaa, A., Chan, A.J., van der Schaar, M.: Generative time-series modeling with fourier flows. In: International Conference on Learning Representations (2021)
2. Alcaraz, J.M.L., Strodthoff, N.: Diffusion-based time series imputation and forecasting with structured state space models. arXiv preprint arXiv:2208.09399 (2022)
3. Brophy, E., De Vos, M., Boylan, G., Ward, T.: Estimation of continuous blood pressure from PPG via a federated learning approach. Sensors **21**(18), 6311 (2021)
4. Ceneda, D., Gschwandtner, T., Miksch, S., Tominski, C.: Guided visual exploration of cyclical patterns in time-series. In: Proceedings of the IEEE Symposium on Visualization in Data Science (VDS). IEEE Computer Society (2018)

5. Desai, A., Freeman, C., Wang, Z., Beaver, I.: Timevae: a variational auto-encoder for multivariate time series generation. arXiv preprint arXiv:2111.08095 (2021)
6. Fons, E., Sztrajman, A., El-Laham, Y., Iosifidis, A., Vyetrenko, S.: Hypertime: implicit neural representation for time series. arXiv preprint arXiv:2208.05836 (2022)
7. Goodfellow, I., et al.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
8. Heckbert, P.: Fourier transforms and the fast fourier transform (FFT) algorithm. Comput. Graph. **2**(1995), 15–463 (1995)
9. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol. 33, pp. 6840–6851 (2020)
10. Jeha, P., et al.: PSA-GAN: progressive self attention GANs for synthetic time series. In: The Tenth International Conference on Learning Representations (2022)
11. Kingma, D.P.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
12. Kitagawa, G., Gersch, W.: A smoothness priors-state space modeling of time series with trend and seasonality. J. Am. Stat. Assoc. **79**(386), 378–389 (1984)
13. Kollovieh, M., Ansari, A.F., Bohlke-Schneider, M., Zschiegner, J., Wang, H., Wang, Y.B.: Predict, refine, synthesize: self-guiding diffusion models for probabilistic time series forecasting. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
14. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
15. Liao, S., Ni, H., Szpruch, L., Wiese, M., Sabate-Vidales, M., Xiao, B.: Conditional sig-Wasserstein GANs for time series generation. arXiv preprint arXiv:2006.05421 (2020)
16. Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. Phil. Trans. R. Soc. A **379**(2194), 20200209 (2021)
17. Liu, Y., et al.: Vertical federated learning: concepts, advances, and challenges. IEEE Trans. Knowl. Data Eng. (2024)
18. Luu, K., Khashabi, D., Gururangan, S., Mandyam, K., Smith, N.A.: Time waits for no one! analysis and challenges of temporal misalignment. arXiv preprint arXiv:2111.07408 (2021)
19. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
20. Meijer, C., Huang, J., Sharma, S., Lazovik, E., Chen, L.Y.: Ts-inverse: a gradient inversion attack tailored for federated time series forecasting models. In: 2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML), pp. 110–124. IEEE (2025)
21. Mendieta, M., Yang, T., Wang, P., Lee, M., Ding, Z., Chen, C.: Local learning matters: rethinking data heterogeneity in federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8397–8406 (2022)
22. Mogren, O.: C-RNN-GAN: continuous recurrent neural networks with adversarial training. arXiv preprint arXiv:1611.09904 (2016)
23. Pratama, I., Permanasari, A.E., Ardiyanto, I., Indrayani, R.: A review of missing values handling methods on time-series data. In: 2016 International Conference on Information Technology Systems and Innovation (ICITSI), pp. 1–6. IEEE (2016)

24. Qu, L., et al.: Rethinking architecture design for tackling data heterogeneity in federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10061–10071 (2022)
25. Rasouli, M., Sun, T., Rajagopal, R.: FedGAN: federated generative adversarial networks for distributed data. arXiv preprint arXiv:2006.07228 (2020)
26. Rasul, K., Seward, C., Schuster, I., Vollgraf, R.: Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In: International Conference on Machine Learning, pp. 8857–8868. PMLR (2021)
27. Sachdeva, N., McAuley, J.: Data distillation: a survey. arXiv preprint arXiv:2301.04272 (2023)
28. Shankar, A., Brouwer, H., Hai, R., Chen, L.: Silofuse: cross-silo synthetic data generation with latent tabular diffusion models (2024)
29. Song, J., Ye, J.C.: Federated cycleGAN for privacy-preserving image-to-image translation. arXiv preprint arXiv:2106.09246 (2021)
30. Tashiro, Y., Song, J., Song, Y., Ermon, S.: CSDI: conditional score-based diffusion models for probabilistic time series imputation. In: Advances in Neural Information Processing Systems, vol. 34, pp. 24804–24816 (2021)
31. Vaswani, A.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017)
32. Voigt, P., Von dem Bussche, A.: The EU general data protection regulation (GDPR). A Practical Guide, 1st ed., pp. 10–5555. Springer International Publishing, Cham (2017)
33. Wang, Z., Cheng, X., Su, S., Wang, G.: Differentially private generative decomposed adversarial network for vertically partitioned data sharing. Inf. Sci. **619**, 722–744 (2023)
34. Yoon, J., Jarrett, D., Van der Schaar, M.: Time-series generative adversarial networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
35. Yuan, X., Qiao, Y.: Diffusion-TS: interpretable diffusion for general time series generation. arXiv preprint arXiv:2403.01742 (2024)
36. Yuan, X., Zhao, Z., Gope, P., Sikdar, B.: VFLGAN-TS: vertical federated learning-based generative adversarial networks for publication of vertically partitioned time-series data. arXiv preprint arXiv:2409.03612 (2024)
37. Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y.: A survey on federated learning. Knowl.-Based Syst. **216**, 106775 (2021)
38. Zhao, Z., Wu, H., Van Moorsel, A., Chen, L.Y.: GTV: generating tabular data via vertical federated learning. arXiv preprint arXiv:2302.01706 (2023)
39. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2223–2232 (2017)