SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2026, Vol. 69, Iss. 5, 152102:1–152102:15
https://doi.org/10.1007/s11432-025-4553-x

Tackling data heterogeneity in federated time series forecasting

Wei YUAN[1], Chaoqun YANG[2], Xiangyu ZHAO[3], Quoc Viet Hung NGUYEN[2], Yang CAO[4], Tieke HE[5] & Hongzhi YIN[1*]

[1]*School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane 4000, Australia*
[2]*School of Information and Communication Technology, Griffith University, Gold Coast 4222, Australia*
[3]*Department of Data Science, City University of Hong Kong, Hong Kong 999077, China*
[4]*Department of Computer Science, Institute of Science Tokyo, Tokyo 145-0061, Japan*
[5]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*

Received 25 April 2025/Revised 27 June 2025/Accepted 12 August 2025/Published online 8 April 2026

Abstract Time series forecasting plays a critical role in various real-world applications, including energy consumption prediction, disease transmission monitoring, and weather forecasting. Although substantial progress has been made in time series forecasting, most existing methods rely on a centralized training paradigm, where large amounts of data are collected from distributed devices (e.g., sensors, wearables) to a central cloud server. However, this paradigm has overloaded communication networks and raised privacy concerns. Federated learning, a popular privacy-preserving technique, enables collaborative model training across distributed data sources. However, directly applying federated learning to time series forecasting often yields suboptimal results, as time series data generated by different devices is inherently heterogeneous. In this paper, we propose a novel framework, federated time series forecasting with synthetic data (Fed-TREND), to address data heterogeneity by generating informative synthetic data as auxiliary knowledge carriers. Specifically, Fed-TREND generates two types of synthetic data. The first type of synthetic data captures the representative distribution information from clients' uploaded model updates and enhances clients' local training consensus. The second kind of synthetic data extracts long-term influence insights from global model update trajectories and is used to refine the global model after aggregation. Fed-TREND is compatible with most time series forecasting models and can be seamlessly integrated into existing federated learning frameworks to improve prediction performance. Extensive experiments on eight datasets, using several federated learning baselines and four popular time series forecasting models, demonstrate the effectiveness and generalizability of Fed-TREND.

Keywords federated learning, time series forecasting, data heterogeneity

Citation Yuan W, Yang C Q, Zhao X Y, et al. Tackling data heterogeneity in federated time series forecasting. Sci China Inf Sci, 2026, 69(5): 152102, https://doi.org/10.1007/s11432-025-4553-x

1 Introduction

With the proliferation of sensors, wearables, and Internet of Things (IoT) devices, the volume of time series data [1,2] has increased dramatically in recent years. Time series forecasting has emerged as a focal point for both academic and industrial communities, reflecting its importance in automatically extracting meaningful patterns from extensive historical data to predict future values. Existing forecasting methods primarily aim to enhance prediction accuracy by employing advanced deep learning architectures to model temporal dependencies [3]. For example, several studies [4,5] have refined the Transformer architecture [6] to handle long-sequence time series prediction efficiently. Meanwhile, recent research [7–9] has explored the use of MLPs for capturing temporal information, achieving state-of-the-art performance.

While the aforementioned models have achieved significant success, most rely on centralized training, where large volumes of data are collected from widely deployed devices and uploaded to a central server or cloud. However, collecting data from distributed devices presents practical challenges due to limited bandwidth and stringent privacy regulations (e.g., GDPR[1)] and CCPA[2)]). For example, electricity usage data can reveal highly sensitive information about individuals, causing data owners to hesitate to share it due to privacy concerns [10]. Similarly, smart wearables record detailed personal health metrics, such as oxygen saturation, heart rate, electrocardiogram (ECG),

* Corresponding author (email: h.yin1@uq.edu.au)
1) https://gdpr-info.eu/.
2) https://oag.ca.gov/privacy/ccpa.

and electroencephalogram (EEG). Given the sensitive nature of this data, sharing it significantly heightens the risk of data breaches.

Federated learning [11, 12] offers a privacy-preserving framework for training predictive models without sharing or transmitting raw data. In this approach, a central server coordinates multiple clients, enabling them to collaboratively train models on their locally stored data. This ensures data privacy while minimizing the need to transmit large volumes of raw data. In this work, we focus on cross-device federated time series forecasting, where each device operates as an individual client due to its growing computational capabilities. This federated framework provides robust privacy protection by ensuring that data remains stored locally on each device. Throughout the subsequent sections, the terms "client" and "device" are used interchangeably. However, traditional federated learning methods assume that data across clients follow independent and identical distributions (IID), which is rarely true in time series data. Time series data are inherently heterogeneous, as they are generated by diverse devices operating under varying conditions and serving various applications. Existing federated learning methods face significant challenges in effectively learning from such heterogeneous data [13].

Two key scenarios give rise to heterogeneity in federated time series forecasting. (1) Multivariate time series forecasting. This scenario occurs when different variables or dimensions of the same entities are collected and stored on separate devices. These variables are inherently heterogeneous and correlated. However, leveraging and integrating their correlations can significantly enhance forecasting performance across all variables, as evidenced by numerous multivariate time series prediction models developed using traditional centralized approaches [14, 15]. In federated time series forecasting, models are trained independently on each variable, with model aggregation serving as the sole mechanism for information sharing. Unfortunately, this approach fails to capture the complex relationships between variables and can even degrade forecasting accuracy for individual variables after aggregation. (2) Heterogeneous temporal patterns. This scenario arises when different devices monitor the same variables or dimensions across distinct entities. For example, electricity usage data collected by smart meters varies significantly across households due to differences in lifestyle and living activities. Aggregating models that capture heterogeneous temporal patterns or distributions often produces a suboptimal global model. To sum up, addressing these heterogeneity issues requires that clients be able to learn relevant information from one another, and that the aggregated model be refined to account for deviations caused by heterogeneous data distributions.

Based on the above analysis, we propose a novel method federated time series forecasting with synthetic data (Fed-TREND). Fed-TREND draws inspiration from recent advancements in data condensation [16–18], where synthetic data is generated from model trajectories to encapsulate the essential information. Fed-TREND generates two types of synthetic data on the central server to enable (1) clients to learn from each other's representative information and (2) to optimize the aggregated global model, therefore, tackling the heterogeneity problems during the federated learning process. Specifically, the first type of synthetic data, $\mathcal{D}_{ct}$, is generated based on model updates uploaded by all clients, encapsulating the representative information of all clients. This synthetic data is then distributed back to each client to augment their local training alongside their own data. By doing so, each client can benefit from all the other clients' variable representative information, akin to training multivariable time series models in a centralized manner. The second type of synthetic data, $\mathcal{D}_{gt}$, is derived from global model trajectories, capturing the dynamic patterns in the global model parameters. This data is used in conjunction with client-uploaded model updates to refine the aggregation of the global model, mitigating the model drift challenges posed by heterogeneous temporal patterns.

Noted that unlike other data condensation-based federated learning approaches [19–22], where synthetic data acts as the primary information carrier for collaborative learning (i.e., the global model is solely trained on the gathered synthetic data), Fed-TREND uses synthetic data as auxiliary information. The primary learning process remains rooted in sharing model parameters and updates, ensuring system performance does not overly depend on the quality of synthetic data, a factor that is often difficult to guarantee. Furthermore, the synthetic data construction occurs entirely on the central server, minimizing the computational burden on client devices.

In conclusion, the major contributions of this work are as follows.

- To the best of our knowledge, we are the first to propose and conduct a comprehensive investigation into the data heterogeneity challenge in cross-device federated time series forecasting.
- We propose a versatile federated time series forecasting component, Fed-TREND, which addresses the challenges of learning from heterogeneous time series data by constructing two types of synthetic data derived from clients' uploaded models and the aggregated global models.
- To validate the effectiveness and generalizability of Fed-TREND, we conducted extensive experiments on eight widely used time series forecasting datasets. By integrating Fed-TREND into several mainstream federated learning frameworks, we train four widely adopted state-of-the-art time series forecasting models: DLinear [8], LightTS [7], TSMixer [9], and iTransformer [5]. The results consistently demonstrate that Fed-TREND significantly improves

the federated forecasting performance across all time series forecasting datasets from diverse application scenarios.

2 Related work

2.1 Time series forecasting

In recent years, deep learning-based approaches have become the dominant trend in time series forecasting [23–25]. These methods typically use neural network architectures, such as RNNs [26, 27], CNNs [28], Transformers [6], and MLPs, as backbone models. For example, LSTNet [29] and TPR-LSTM [30] combine CNNs and RNNs with attention mechanisms to capture both short- and long-term dependencies in time series data. Transformers have recently gained popularity in time series forecasting due to their ability to model global dependencies [31]. Zhou et al. [4] introduced Informer, which reduces time complexity and enhances memory efficiency, while Liu et al. [5] innovatively reversed the data dimensions in the Transformer's attention and feed-forward layers. The use of pure multilayer perceptron (MLP) in time series forecasting has also become a recent trend because of their implementation simplicity and performance effectiveness, such as DLinear [8], LightTS [7], and TSMixer [9]. However, all of these methods are implemented in a centralized manner, overlooking the practical challenges of data privacy in real-world applications.

2.2 Federated learning with data heterogeneity

Federated learning enables collaborative training of a global model without requiring access to clients' raw data. This learning paradigm has garnered significant attention for applications where data collection is challenging [32]. FedAvg [11] was the first and remains the most widely used federated learning framework. It trains a global model by averaging the local models of participating clients. However, FedAvg's performance suffers when data across clients are heterogeneous, a common situation as clients independently collect data in diverse environments.

To address data heterogeneity in federated learning, numerous methods have been proposed [12, 13]. Broadly, these approaches can be divided into two categories based on whether they maintain compatibility with the original FedAvg protocol. For methods incompatible with FedAvg, additional assumptions or altered learning pipelines are typically required. For example, some approaches assume the central server has access to public data [33–35] or that data transmission between clients is allowed [36, 37]. These extra requirements limit their practical applicability. Therefore, in this paper, we focus on approaches to handling data heterogeneity within the standard federated learning framework. FedProx [38] introduces a proximal term during local model training to prevent local updates from deviating too far from the global model. SCAFFOLD [39] employs a control variate for variance reduction to stabilize aggregation. FedDyn [40] uses dynamic regularization to adjust each client's objective during training. Elastic [41] designs an elastic aggregation approach that dampens the influence of updates to sensitive parameters. Chen et al. [42] proposed FedHEAL, incorporating a fair aggregation objective to prevent global model bias toward specific domains.

Unfortunately, most of these studies address data heterogeneity in image classification tasks. Our empirical results reveal that these approaches fail to achieve significant improvements in federated time series forecasting. This is due to the unique nature of heterogeneity in time series forecasting, which differs fundamentally from that in traditional image classification. In image classification tasks, heterogeneity typically stems from variations in label or domain distributions across clients. In contrast, heterogeneity in time series forecasting arises from differences in variable types and the complex, evolving temporal patterns of the time series data.

2.3 Federated learning with data condensation

Data condensation aims to compress a large training dataset into a smaller, synthetic dataset [17] and has recently been integrated into federated learning [18]. This integration serves two primary purposes: (1) to improve communication efficiency [43–46] and (2) to address data heterogeneity [19–22, 47]. Here, we focus on the latter. Goetz et al. [19] and Xiong et al. [20] proposed a standard workflow where clients locally compress a small synthetic dataset and share it with the central server. The server then trains a global model on the gathered synthetic data and distributes this model back to the clients. Wang et al. [22] extended this approach by allowing clients to upload average logits of real data, further improving system performance. However, these methods have several limitations. First, their performance heavily relies on the quality of the synthetic data generated by each client, which is difficult to guarantee. Additionally, these approaches require clients to have substantial computational resources, as generating synthetic data is computationally intensive. In federated time series forecasting, clients are often sensors or mobile devices with limited computational capacity, making these methods less suitable for such environments.

Table 1 List of important notations.

Notation	Description
c_i	A client/device in federated time series forecasting
$\mathcal{D}_{c_i}$	The local dataset for client c_i
$\mathcal{D}_{ct}$, $\mathcal{D}_{gt}$	Synthetic dataset generated using clients'/global model trajectories
$\mathcal{T}_{ct}$, $\mathcal{T}_{gt}$	Trajectories bank for client/global model updates
$\mathbf{X}_j^{c_i}$, $\mathbf{Y}_j^{c_i}$	The jth input/target output data for client c_i
$\mathbf{X}_i^{ct}$, $\mathbf{Y}_i^{ct}$	The ith input/target output data (trainable parameters) in $\mathcal{D}_{ct}$
$\mathbf{X}_i^{gt}$, $\mathbf{Y}_i^{gt}$	The ith input/target output data (trainable parameters) in $\mathcal{D}_{gt}$
$\mathbf{W}_{c_i}^t$, $\mathbf{W}^t$	The model trained by c_i/the aggregated global model at round t
L_x, L_y	The input/output data length
L_{ct}	The update frequency of $\mathcal{D}_{ct}$ and the trajectories segment length in $\mathcal{D}_{ct}$ construction
L_{gt}	The update frequency of $\mathcal{D}_{gt}$
L_{gt}^{seg}	The trajectories segment length in $\mathcal{D}_{gt}$ construction

DynaFed [47] is more closely related to our approach, Fed-TREND, but it only uses synthetic data to adjust the global model and is designed for image classification tasks.

2.4 Federated learning in time series forecasting

The research on federated time series forecasting is still under-explored. Time-FFM [48] investigates this topic at the organization level, where each data organization (e.g., traffic data organization or electrical data organization) acts as a client. Abdel-Sater et al. [49] applied organization-level federated learning to train a time series forecasting model based on large language models. Yan et al. [50] proposed a vertical federated learning structure. In this paper, we propose a device-level federated time series forecasting framework that alleviates data heterogeneity by generating synthetic data.

3 Preliminaries

In this section, we present a formal introduction to the settings of federated time series forecasting and then briefly introduce the basic time series forecasting models. Note that, we use squiggle uppercase (e.g., $\mathcal{A}$) to indicate sets or algorithms, bold lowercase (e.g., $\mathbf{a}$) to represent vectors, and bold uppercase (e.g., $\mathbf{A}$) to denote matrices or tensors. Table 1 lists some important notations.

3.1 Formulation of federated time series forecasting

Let $\mathcal{C} = \{c_i\}_{i=1}^{|C|}$ be the set of clients/devices and $|C|$ is the number of all clients. For a client c_i, it owns time series data $\mathbf{X}_{c_i} = [\mathbf{x}_1^{c_i}, \mathbf{x}_2^{c_i}, \ldots, \mathbf{x}_{T-1}^{c_i}, \mathbf{x}_T^{c_i}] \in \mathbb{R}^{T_{c_i} \times f_{c_i}}$, where T_{c_i} is the total length of the data and f_{c_i} is the number of dimensions. Notably, in federated time series forecasting, the time series data $\mathbf{X}_{c_i}$ is always kept on the corresponding device and can not be accessed by any other participants. To train a time series forecasting model, clients construct a dataset $\mathcal{D}_{c_i} = \{(\mathbf{X}_j^{c_i}, \mathbf{Y}_j^{c_i})\}_{j=1}^{|\mathcal{D}_{c_i}|}$ based on $\mathbf{X}_{c_i}$, where $\mathbf{X}_j^{c_i} = [\mathbf{x}_j^{c_i}, \mathbf{x}_{j+1}^{c_i}, \ldots, \mathbf{x}_{j+L_x-1}^{c_i}, \mathbf{x}_{j+L_x}^{c_i}] \in \mathbb{R}^{L_x \times f}$ is a fragment of time series data as the input of a forecasting model $\mathcal{F}(\cdot)$ and $\mathbf{Y}_j^{c_i} = [\mathbf{x}_{j+L_x+1}^{c_i}, \mathbf{x}_{j+L_x+2}^{c_i}, \ldots, \mathbf{x}_{j+L_x+L_y-1}^{c_i}, \mathbf{x}_{j+L_x+L_y}^{c_i}] \in \mathbb{R}^{L_y \times f}$ is the target future prediction. Then, the goal of federated time series forecasting can be described as

$$\underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{|\mathcal{C}|} \sum_{c_i \in \mathcal{C}} \frac{1}{|\mathcal{D}_{c_i}|} \mathcal{L}(\mathbf{W}, \mathcal{D}_{c_i}), \tag{1}$$

$$\mathcal{L}(\mathbf{W}, \mathcal{D}) = \sum_{(\mathbf{X}_j, \mathbf{Y}_j) \in \mathcal{D}} \left\| \mathbf{Y}_j - \mathcal{F}(\mathbf{W}, \mathbf{X}_j) \right\|, \tag{2}$$

where $\mathbf{W}$ is the parameters of the forecasting model.

Federated time series forecasting employs a central server to coordinate clients to optimize (1) without accessing clients' distributed datasets $\mathcal{D}_{c_i}$ by transmitting and aggregating model parameters. Specifically, clients and the central server iteratively repeat the following steps until model convergence. At the round of t, a central server selects a group of clients $\mathcal{C}^t$ to participate in the training process and disperses the global model parameters $\mathbf{W}^t$ to them. Subsequently, clients leverage the received global model parameters to initialize a local model and optimize the local

model with (2) on their local datasets $\mathcal{D}_{c_i}$. After local training, clients upload the updated model parameters $\mathbf{W}_{c_i}^t$ to the central server. When the updated parameters are received, the central server aggregates these parameters to form new global model parameters. One main-stream aggregation solution is FedAvg [11], which utilizes a weighted average to aggregate client-uploaded parameters. This design performs well when the client data is homogeneous. However, when client data is heterogeneous, local models are updated towards a local optimal solution, and FedAvg cannot simply aggregate them to achieve optimal global performance.

3.2 Base time series forecasting models

A federated time series forecasting framework should ideally be compatible with most time series forecasting models. In this paper, to demonstrate the generalizability of our proposed method, we select four recent state-of-the-art time series models that cover two major architectures: Transformer (iTransformer [9]) and MLP (DLinear [8], LightTS [7], and TSMixer [9]), as these models are the existing state-of-the-art and representative centralized time series forecasting models.

4 Methodology

4.1 Overview of Fed-TREND

In Section 1, we analyze two key scenarios in federated time series forecasting: (1) the time series data on clients correspond to different variables, and (2) clients have the same variables but with distinct temporal patterns due to their unique characteristics. Based on this analysis, we identify two critical weaknesses of the original federated learning framework that hinder its ability to handle such heterogeneous scenarios. First, clients rely solely on model aggregation for knowledge transfer, which fails to capture the complex relationships between clients, especially when these clients correspond to different variables. Second, the server lacks the capability to aggregate a superior global model from the uploaded client models, particularly when faced with heterogeneous data distributions.

In light of this, we introduce Fed-TREND, a framework designed to address heterogeneity in federated time series forecasting by improving knowledge transfer among clients while calibrating a better aggregated global model. Fed-TREND achieves these objectives by constructing two types of synthetic data using model trajectories from various sources. The first type of synthetic data, $\mathcal{D}_{ct}$, encapsulates the representative distribution information derived from clients' uploaded model updates. This data acts as a "proxy" that enhances knowledge transfer by integrating with clients' local training. The second type of synthetic data, $\mathcal{D}_{gt}$, captures the long-term dynamic changes in the aggregated global model trajectories and is used to refine the global model. An overview of Fed-TREND is depicted in Figure 1, and its workflow is presented as pseudocode in Algorithm 1.

Notably, our Fed-TREND is compatible with most federated learning frameworks as it addresses heterogeneity from the synthetic data construction aspect and does not break the basic federated learning protocol. Without loss of generality, we introduce Fed-TREND based on the most general learning framework illustrated in Subsection 3.1. In the experimental section, we also evaluate the empirical performance of Fed-TREND when integrated with various federated learning frameworks.

4.2 Synthetic data $\mathcal{D}_{ct}$ for representative knowledge transfer

The recently developed technique of data condensation [16] has demonstrated that a synthetic dataset can be learned from model training trajectories to summarize useful information. Building on this idea, Fed-TREND introduces a synthetic dataset, $\mathcal{D}_{ct}$, designed to capture representative knowledge of all clients from clients' uploaded model parameters. Specifically, at each federated training round t, when client c_i uploads its updated model parameters $\mathbf{W}_{c_i}^t$, except for using this model update for model aggregation, the central server will also store them in a trajectories bank $\mathcal{T}_{ct} = \{c_i : [\mathbf{W}_{c_i}^1, \ldots, \mathbf{W}_{c_i}^t]\}_{c_i \in \mathcal{C}}$. Then, the central server will optimize a synthetic dataset $\mathcal{D}_{ct}$ based on $\mathcal{T}_{ct}$ as follows:

$$\underset{\mathcal{D}_{ct}}{\operatorname{argmin}} \mathbb{E}_{c_j \sim U(\mathcal{C}), s \sim U(1, t-L_{ct})} \left[d(\widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}}, \mathbf{W}_{c_j}^{s+L_{ct}}) \right], \tag{3}$$

$$\text{s.t.} \widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}} = \operatorname{argmin} \mathcal{L}(\mathbf{W}_{c_j}^{s}, \mathcal{D}_{ct}). \tag{4}$$

Here, L_{ct} is the segment length of a trajectory and $U(\cdot)$ denotes a uniform random sampling. The meaning of the above two formulas is that, in (4), we train a forecasting model initialized from $\mathbf{W}_{c_j}^s$ on $\mathcal{D}_{ct} = \{(\mathbf{X}_i^{ct}, \mathbf{Y}_i^{ct})\}_{i=1}^{|\mathcal{D}_{ct}|}$ for L_{ct} steps, obtaining the trained model $\widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}}$. Note that same as the settings in the real dataset, $\mathbf{X}_i^{ct}$ and $\mathbf{Y}_i^{ct}$

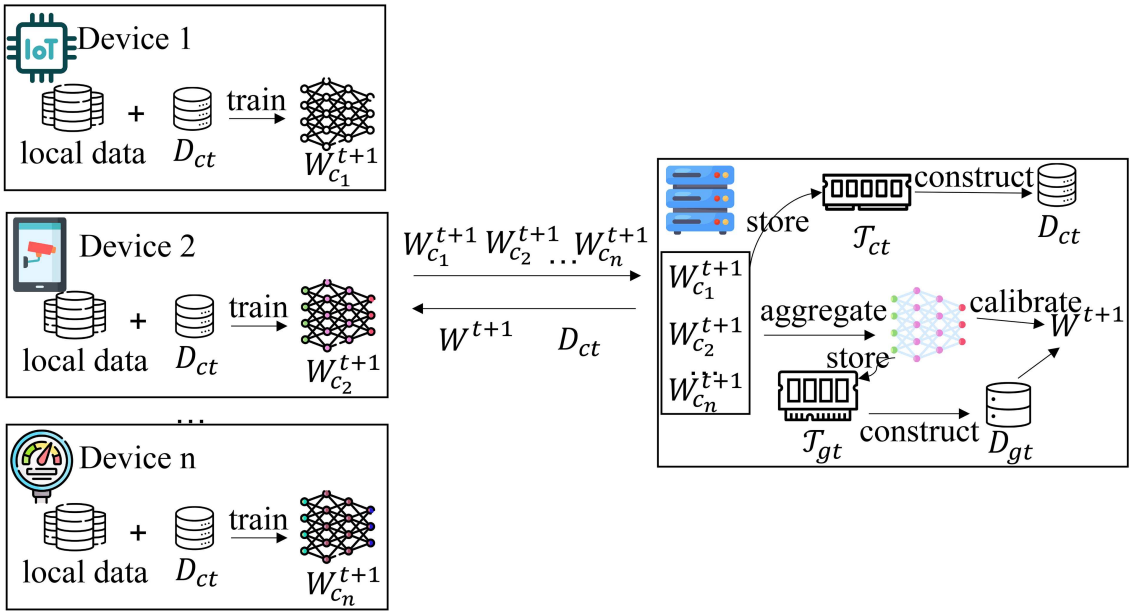

Figure 1 (Color online) The overall architecture of Fed-TREND. When clients upload their model updates, these updates are (1) used for aggregation as the original federated learning and (2) stored in a trajectory bank $\mathcal{T}_{ct}$ for $\mathcal{D}_{ct}$ construction. Besides, the aggregated global model is recorded in another trajectory bank $\mathcal{T}_{gt}$, which is used to construct the synthetic data $\mathcal{D}_{gt}$. After that, $\mathcal{D}_{ct}$ is sent back to clients for local training, while $\mathcal{D}_{gt}$ is used to refine the aggregated global model.

are the input and target output pair. Then, in (3), we minimize the distance between $\widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}}$ and $\mathbf{W}_{c_j}^{s+L_{ct}}$ via optimizing $\mathcal{D}_{ct}$, i.e., the synthetic time series data pair $(\mathbf{X}_i^{ct}, \mathbf{Y}_i^{ct})$ is learnable parameters.

Intuitively, by optimizing (3) and (4), we can obtain a synthetic dataset $\mathcal{D}_{ct}$, where initializing a model with any client's model updates from $\mathcal{T}ct$ and subsequently training the model on $\mathcal{D}_{ct}$ produces updates similar to those obtained by training on the clients' original local data. In other words, $\mathcal{D}_{ct}$ effectively captures the essential information of all clients' local data for training their local models from initialization to round t.

Ideally, the optimization should be performed every time that clients upload new model updates. However, constructing the synthetic dataset is computationally intensive due to its bi-level optimization process, posing a heavy burden on the central server. Since the goal of Fed-TREND to construct $\mathcal{D}_{ct}$ is to learn the representative information among clients rather than replacing the original dataset in each client, as is the traditional data condensation goal, which requires a very high quality of synthetic data, we simplify the process to reduce computational and memory costs. In detail, we only update the synthetic datasets at intervals of L_{ct}. For each client c_i, the central server temporarily stores only the start and end model updates within these intervals $\mathbf{W}_{c_j}^{k*L_{ct}}$ and $\mathbf{W}_{c_j}^{(k+1)*L_{ct}}$, i.e., $\mathcal{T}_{ct} = \{c_i : (\mathbf{W}_{c_j}^{k*L_{ct}}, \mathbf{W}_{c_j}^{(k+1)*L_{ct}})\}_{c_i \in \mathcal{C}}$. Then, when the round $t = (k+1) * L_{ct}$, the central server will construct a synthetic dataset $\mathcal{D}_{ct}^t$ based on the trajectories $\mathcal{T}_{ct}$. That is to say, (3) is simplified to (5):

$$\underset{\mathcal{D}_{ct}}{\operatorname{argmin}}\, \mathbb{E}_{c_j \sim U(\mathcal{C})} \left[d(\widetilde{\mathbf{W}}_{c_j}^{k*L_{ct}}, \mathbf{W}_{c_j}^{(k+1)*L_{ct}}) \right]. \tag{5}$$

After obtaining the optimized $\mathcal{D}_{ct}$, $\mathcal{T}_{ct}$ is set to empty and used to record the next pair of start and end parameters of L_{ct} length model trajectories. Therefore, it can also reduce the memory burden as we only need to the store a pair of model updates for clients, rather than clients' whole model updates.

Moreover, to further reduce the optimization difficulty and make the synthetic dataset focus on learning significant information from trajectories $\mathcal{T}_{ct}$, we only utilize the model parameters that consistently update towards the same directions as the learning sources for $\mathcal{D}_{ct}^t$ optimization. According to [42], the parameters that consistently update towards a direction may reflect some important signals, and learning from these parameters can make the synthetic dataset more concentrated on extracting this knowledge. Therefore, before the parameters $\widetilde{\mathbf{W}}_{c_j}^{k*L_{ct}}$ are added to the trajectories memory bank $\mathcal{T}_{ct}$, the central server will first check whether the gradient of the element $w_{c_j,m} \in \widetilde{\mathbf{W}}_{c_j}^{k*L_{ct}}$ is consistent with its previous updates, i.e., $\text{sign}(\Delta w_{c_j,m}^{k*L_{ct}-1}) == \text{sign}(\Delta w_{c_j,m}^{k*L_{ct}})$. If the update is not consistent, the distance loss of the corresponding element in (5) will be masked.

After the central server constructs the synthetic dataset $\mathcal{D}_{ct}$, it is dispersed to each client and mixed with the clients' local dataset for local training. Since the dataset $\mathcal{D}_{ct}$ is optimized on the consistent model trajectories from

all clients, it captures the representative knowledge of all clients' local data. Thus, incorporating this synthetic dataset helps mitigate local data heterogeneity, enabling clients to learn from each other indirectly.

Algorithm 1 The pseudo-code for the Fed-TREND.

Input: Global round R; learning rate lr, L_{ct}, L_{gt}, ...
Output: Well-trained time series forecasting model $\mathbf{W}^R$.

Server initializes model $\mathbf{W}^0$;
$\mathcal{T}_{ct} = \{c_i : [\mathbf{W}^0_{c_i}]\}_{c_i \in \mathcal{C}}$, $\mathcal{T}_{gt} = \{\mathbf{W}^0\}$;
$\mathcal{D}_{ct} = \emptyset, \mathcal{D}_{gt} = \emptyset$;
for Each round $t = 0, \ldots, R-1$ **do**
  Sample a fraction of clients $\mathcal{C}^t$ from $\mathcal{C}$;
  for $c_i \in \mathcal{C}^t$ **in parallel do**
    // Execute on client sides;
    Download $\mathbf{W}^t$ and $\mathcal{D}_{ct}$;
    $\mathbf{W}^{t+1}_{c_i} \leftarrow$ update local model with forecasting objective (2) on $\mathcal{D}_{c_i}$ and $\mathcal{D}_{ct}$;
    Upload $\mathbf{W}^{t+1}_{c_i}$;
  end for
  // Execute on central server;
  if $t\%L_{ct} == 0$ **then**
    Check each element's update direction consistency;
    Append $\mathbf{W}^{t+1}_{c_i}$ into $\mathcal{T}_{ct}$;
  end if
  $\mathbf{W}^{t+1} \leftarrow$ aggregate received client model parameters $\{\mathbf{W}^{t+1}_{c_i}\}_{c_i \in \mathcal{C}^t}$;
  Append $\mathbf{W}^{t+1}$ into $\mathcal{T}_{gt}$;
  Refine $\mathbf{W}^{t+1}$ on $\mathcal{D}_{gt}$;
end for
if $t\%L_{ct} == 0$ **then**
  $\mathcal{D}_{ct} \leftarrow$ Algorithm 2;
end if
if $t\%L_{gt} == 0$ **then**
  $\mathcal{D}_{gt} \leftarrow$ Algorithm 2;
end if

Algorithm 2 The pseudo-code for the synthetic data construction.

Input: Global round R; learning rate lr, L_{ct}, L_{gt}, ...
Output: Well-trained synthetic data $\mathcal{D}_{syn}$.

Randomly initialize synthetic dataset $\mathcal{D}_{syn}$;
for Synthetic data training iteration $n = 1, \ldots, N$ **do**
  Sample a segment of trajectories $(\mathbf{W}^{start}, \mathbf{W}^{end})$ from trajectories bank $\mathcal{T}$;
  $\widetilde{\mathbf{W}}^{end} \leftarrow$ train $\mathbf{W}^{start}$ on $\mathcal{D}_{syn}$;
  Compute distance loss $d(\widetilde{\mathbf{W}}^{end}, \mathbf{W}^{end})$ and gradient based on $\mathcal{D}_{syn}$;
end for

4.3 Synthetic data for global model refinement

Although the synthetic data $\mathcal{D}_{ct}$ can carry some representative and relationship information from all clients to improve the local training consensus, the heterogeneity problem may still persist. For example, clients' local data still contain personalized temporal patterns that are heterogeneous since clients are naturally distributed in different environments recording different entities. Consequently, the global model may still drift away from the optimal point after aggregating these local models that are learned with various patterns. To address this, Fed-TREND introduces an additional synthetic dataset $\mathcal{D}_{gt}$ to refine the aggregated global model.

Specifically, Fed-TREND constructs $\mathcal{D}_{gt}$ based on the trajectories of aggregated global models, so that the dataset can capture the long-term dynamics of mutual influences of clients' models trained on their heterogeneous local data. Formally, the central server maintains a trajectory bank $\mathcal{T}_{gt} = \{\mathbf{W}^1, \ldots, \mathbf{W}^t\}$, storing the aggregated global model $\mathbf{W}^t$ at each round. Then, $\mathcal{D}_{gt}$ is optimized as follows:

$$\underset{\mathcal{D}_{gt}}{\operatorname{argmin}}\, \mathbb{E}_{s\sim U(1,t-L^{seg}_{gt})}\left[d(\widetilde{\mathbf{W}}^{s+L^{seg}_{gt}}, \mathbf{W}^{s+L^{seg}_{gt}})\right], \text{s.t. } \widetilde{\mathbf{W}}^{s+L^{seg}_{gt}} = \operatorname{argmin} \mathcal{L}(\mathbf{W}^s, \mathcal{D}_{gt}), \tag{6}$$

where L^{seg}_{gt} is the length of the trajectory segment and $\widetilde{\mathbf{W}}^{s+L^{seg}_{gt}}$ is the model trained on $\mathcal{D}_{gt}$ from the initialization point of $\widetilde{W}^s$ for L^{seg}_{gt} steps. Similar to $\mathcal{D}_{ct}$, $\mathcal{D}_{gt}$ is constructed with pairs of learnable input and target outputs $\mathcal{D}_{gt} = \{(\mathbf{X}^{gt}_i, \mathbf{Y}^{gt}_i)\}^{|\mathcal{D}_{gt}|}_{i=1}$ and has been optimized using (6). For computational efficiency, $\mathcal{D}_{gt}$ is updated at intervals of L_{gt} rounds, similar to the update strategy for $\mathcal{D}_{ct}$.

Once constructed, $\mathcal{D}_{gt}$ effectively summarizes the stable influence of clients' data distributions on model aggregation within the recent L_{gt} federated learning rounds. Then, for the future aggregated global model $\mathbf{W}^t$, we refine it by finetuning on $\mathcal{D}_{gt}$ for calibration.

4.4 Implementation of synthetic data construction

We employ the most commonly used synthetic data construction algorithm MTT [16] to construct both $\mathcal{D}_{ct}$ and $\mathcal{D}_{gt}$, considering its state-of-the-art performance for meaningful data construction. Note that Fed-TREND is also compatible with other data construction algorithms, such as distribution DC [51], PP [52], and FTD [53]. The detailed steps for synthetic data construction are outlined in Algorithm 2. We begin by randomly initializing the synthetic data and then optimizing it over several iterations. In each iteration, we first sample a model trajectory and train the model's starting point on the synthetic dataset. After training, we compute the distance between the trained model and the endpoint of the trajectory, and then update the synthetic data to make the trained model more closely resemble the endpoint model.

4.5 Discussion

4.5.1 *Privacy analysis*

First, the synthetic data in Fed-TREND are constructed in compliance with the standard federated learning protocol, without introducing any additional assumptions. Specifically, from the server's perspective, compared to traditional federated learning, it only receives model updates from clients without any additional information. Therefore, if the server is a curious-but-honest adversary, the privacy risk should not increase in Fed-TREND. From the client's perspective, compared to traditional federated learning, each client receives an additional material: a synthetic dataset $\mathcal{D}_{ct}$. However, since $\mathcal{D}_{ct}$ is generated based on updates from all clients, it represents a fusion and aggregation of information across participants, similar to the aggregated updates. Therefore, it should be difficult for the possibly curious-but-honest clients to infer private information about others from this synthetic data. Consequently, the privacy-preserving capability of Fed-TREND remains uncompromised compared to traditional federated learning methods [54]. Moreover, Fed-TREND aligns with existing privacy-preserving methods used in federated learning, such as local differential privacy (LDP). In particular, the implementation of LDP in Fed-TREND mirrors that in standard federated learning: clients simply add LDP noise to the uploaded model parameters. Since the synthetic data are constructed using these perturbed model parameters, considered a form of post-processing [55], the system remains compliant with LDP requirements. In Subsection 5.9, we empirically evaluate the compatibility of Fed-TREND with privacy-preserving mechanisms [56].

4.5.2 *Communication cost analysis*

Fed-TREND introduces some additional communication overhead because the central server needs to distribute $\mathcal{D}_{ct}$ to clients. However, this cost is minimal. Taking our experimental setup as an example, the central server sends $\mathcal{D}_{ct}$ to clients at intervals of 10 rounds, with $\mathcal{D}_{ct}$ consisting of 20 input-output pairs. Therefore, for the entire training process (spanning 80 rounds), the extra communication cost per client is $8 \times 20 \times (size(\mathbf{X}^{ct}) + size(\mathbf{Y}^{ct}))$. Given that $size(\mathbf{X}^{ct})$ is a sequence of 24 numbers in our case, the total additional cost is less than 30 kB per client.

4.5.3 *Computational burden analysis*

Existing condensation techniques often suffer from high computational costs. To mitigate this issue in federated learning systems, unlike other studies [19–22] that require clients to perform synthetic data construction, Fed-TREND offloads the entire synthetic data construction process to the central server. This design choice is based on the fact that in practical applications, clients typically have limited computational resources, whereas the central server usually possesses ample computational power. As a result, the computational burden on clients in Fed-TREND remains unchanged, while the extra load on the central server is manageable, especially considering its substantial resources and the potential for performance improvement.

5 Experiments

In this section, we conduct experiments to answer the following research questions.

RQ1. How effective is our Fed-TREND compared to existing federated learning baselines?

RQ2. How is the generalization ability of our Fed-TREND?

Table 2 The statistics of datasets.

Dataset	Client num	Timesteps	Granularity	Domain	Start time
ETTh1, ETTh2	7	14400	1 h	Energy	2016/7/1
Electricity	321	26304	1 h	Energy	2012/1/1
Traffic	862	17544	1 h	Traffic	2015/1/1
Solar Energy	137	52560	10 min	Energy	2006/1/1
State-ILI	37	345	1 week	Disease	2009/10/4
Country-Temp	131	2277	1 month	Climate	1823/1/1
USWeather	12	35064	1 h	Climate	2010/1/1

RQ3. How does Fed-TREND benefit from each key component?

RQ4. How does the value of some important hyper-parameters (e.g., synthetic data sizes, frequency of synthetic data updates, input and output data lengths) affect Fed-TREND's performance?

RQ5. Further study: how is the performance of Fed-TREND with a privacy protection mechanism?

5.1 Datasets

We validate Fed-TREND on eight datasets (ETTh1, ETTh2, Electricity, Traffic, Solar Energy, State-ILI, Country-Temp, and USWeather), covering four different domains (energy, traffic, disease, and climate forecasting). The statistics of these datasets are listed in Table 2. For each dataset, 70% of the data are used for training and validation, while the remaining 30% are reserved for testing. In this paper, we focus on cross-device federated time series forecasting, where each device (e.g., sensors, meters, and wearables) is treated as a client. Consequently, in ETTh1, ETTh2, and USWeather, clients track different variables, whereas in the remaining datasets, clients record the same variable for different entities.

5.2 Evaluation metrics

Following previous time series forecasting studies [5, 7–9], we use mean squared error (MSE) and mean absolute error (MAE) as the primary metrics to evaluate model performance. MSE measures the average of the squared differences between the predicted and actual values, while MAE calculates the average of the absolute differences between the forecasted and actual values.

5.3 Baselines

To demonstrate the effectiveness of Fed-TREND, we compare it against several baselines, including the traditional centralized method (Centralized), the basic federated learning method (FedAvg), and state-of-the-art solutions for data heterogeneity (FedProx, FedDyn, Elastic, FedHEAL, and DynaFed).

- **Centralized.** Train time series forecasting models on a central server by collecting data from devices.
- **FedAvg [11].** The most widely used federated learning framework that updates the global model using average.
- **FedProx [38].** An extension of FedAvg, FedProx introduces a proximal term in the local training objective to stabilize updates from clients with diverse data distributions.
- **FedDyn [40].** FedDyn addresses data heterogeneity by adding a dynamic regularization term to the local objective function, helping synchronize client updates with the overall federated learning process.
- **Elastic [41].** This approach handles data heterogeneity by selectively weighting or attenuating client updates, ensuring the global model benefits from stable, consistent patterns while minimizing the impact of divergent updates due to local data variations.
- **FedHEAL [42].** A state-of-the-art technique designed to address domain skew, FedHEAL maintains both local consistency and domain diversity, enhancing the model generalization across client domains.
- **DynaFed [47].** DynaFed is designed for image classification. Its synthetic data are similar to one of our synthetic data ($\mathcal{D}_{gc}$) but do not update during training.

5.4 Implementation details

For the main experiments, we use DLinear [8] as the default base model for time series forecasting in both Fed-TREND and the baselines, considering its effectiveness and efficiency. In Subsection 5.6, we further evaluate the performance of other forecasting models within Fed-TREND. We set both the input and output lengths (i.e., L_x and L_y) to 24 and examine the impact of these lengths in Subsection 5.8.3. The total number of global training

Table 3 The comparison of the overall performance of Fed-TREND and baselines. The best values of federated learning methods on each dataset are bold.

Dataset	Electricity		Traffic		Solar energy		State-ILI		Country-Temp		ETTh1		ETTh2		USWeather	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Centralized	0.21822	0.30709	0.47615	0.40192	0.32125	0.43077	0.89061	0.68811	0.26942	0.37543	0.37308	0.40949	0.15794	0.27620	0.45217	0.44744
FedAvg	0.22199	0.31156	0.48622	0.41120	0.32251	0.43210	0.96516	0.72040	0.54606	0.57863	0.39343	0.42228	0.16318	0.28154	0.45444	0.45011
FedProx	0.22288	0.31273	0.48815	0.41281	0.32588	0.43608	0.96614	0.72401	0.54606	0.57983	0.39646	0.42428	0.16524	0.28367	0.45533	0.45123
FedDyn	0.21991	0.31628	0.48023	0.40865	0.33385	0.44509	0.96454	0.72014	0.54044	0.57762	0.38909	0.41953	0.15655	0.27625	0.44625	0.45352
Elastic	0.21450	0.30289	0.47445	0.39996	0.32352	0.43345	0.96387	0.71986	0.63603	0.63226	0.38215	0.41568	0.16165	0.28026	0.45054	0.44734
FedHEAL	0.22061	0.31261	0.48187	0.40621	0.32237	0.43123	0.96489	0.72027	0.53606	0.57324	0.37465	0.41158	0.16188	0.28121	0.45202	0.45350
DynaFed	0.23515	0.33116	0.53152	0.44961	0.32481	0.43468	0.95894	0.71776	0.57391	0.61212	0.41513	0.43578	0.15899	0.27764	0.45613	0.45683
Ours	**0.20888**	**0.29838**	**0.46073**	**0.38698**	**0.31457**	**0.42284**	**0.91795**	**0.70034**	**0.45429**	**0.54099**	**0.35814**	**0.39937**	**0.14449**	**0.26381**	**0.44036**	**0.44247**

rounds R is set to 80, with a local epoch count of 1, and all clients participate in each training round. For local training, we use SGD [57] as the optimizer, with a learning rate of 5×10^{-4} and momentum of 0.9. The batch size for local training is 256. For synthetic data generation, we set the update intervals for $\mathcal{L}_{gt}$ and $\mathcal{L}_{ct}$ to 10, meaning the synthetic datasets $\mathcal{D}_{gt}$ and $\mathcal{D}_{ct}$ are updated every ten global rounds. The influence of these intervals is explored in Subsection 5.8.4. The sizes of $\mathcal{D}_{gt}$ and $\mathcal{D}_{ct}$ are explored in Subsections 5.8.2 and 5.8.1, respectively. The optimizer for synthetic data construction is Adam [58] with a learning rate of 3×10^{-4}, and the number of learning iterations is set to 300, following [16].

5.5 Fed-TREND vs. baselines (RQ1)

To demonstrate the effectiveness of Fed-TREND, we compare it with several baselines in Table 3. The results show that the traditional federated learning framework (FedAvg) often lags behind centralized training. The performance gap varies across datasets due to differing degrees of data heterogeneity. For example, on the Solar Energy dataset, the performance gap between Centralized and FedAvg is very close. This is because the data in this dataset are almost homogeneous since at any plant, the solar energy is zero at night and variations among different areas in a state are minimal. In contrast, on datasets like the State-ILI and Country-Temp datasets, intuitively, the data are highly heterogeneous, as illness statistics differ significantly across states, and ground temperatures vary widely among countries. Consequently, the naive FedAvg approach performs much worse than centralized training in these cases. Furthermore, most existing solutions designed to address heterogeneity in image classification do not perform well in federated time series forecasting. As discussed in the previous section, this is due to fundamental differences in task settings and the nature of data heterogeneity between image classification and time series forecasting. Finally, our proposed method, Fed-TREND, consistently outperforms federated learning baselines by a large margin. Notably, on six datasets (Electricity, Traffic, Solar Energy, ETTh1, ETTh2, and USWeather), Fed-TREND even surpasses the performance of centralized training. This improvement may be attributed to the synthetic datasets, which provide additional insights, helping the model better capture temporal patterns. On the more heterogeneous datasets, State-ILI and Country-Temp, while Fed-TREND still has a performance gap compared to Centralized, it achieves the best results among all federated learning baselines.

5.6 The generalization of Fed-TREND (RQ2)

Beyond its effectiveness, Fed-TREND also offers strong generalization capabilities. In this section, we explore this generalizability from two perspectives: (1) can Fed-TREND enhance existing federated learning frameworks by integrating with them? and (2) can Fed-TREND effectively work with various time series forecasting models?

First, we investigate whether Fed-TREND can improve the performance of federated learning baselines. Specifically, we integrate the construction process of $\mathcal{D}_{gt}$ and $\mathcal{D}_{ct}$ into each federated learning method's central server. After aggregation, $\mathcal{D}_{gt}$ is used to finetune the global model, while $\mathcal{D}_{ct}$ is mixed with local data for local training. Figure 2 shows the performance results of equipping federated baselines with Fed-TREND. According to the results, Fed-TREND enhances the performance of all baselines across all datasets, demonstrating its strong generalization ability within different federated learning frameworks.

Additionally, we test Fed-TREND's compatibility with four time series forecasting models, including two of the most popular architectures: MLP and Transformer. As shown in Table 4, Fed-TREND improves the performance of all tested forecasting models compared to naive federated learning. Specifically, DLinear and TSMixer achieve the best performance among the four models, while iTransformer performs the worst. This is likely because, in our cross-device federated time series forecasting setting, each device tracks only a single variable. Therefore, iTransformer, which is designed to fuse information across multiple variables, is less effective in this context. In conclusion, Fed-TREND demonstrates strong generalizability across both federated learning frameworks and various time series forecasting models.

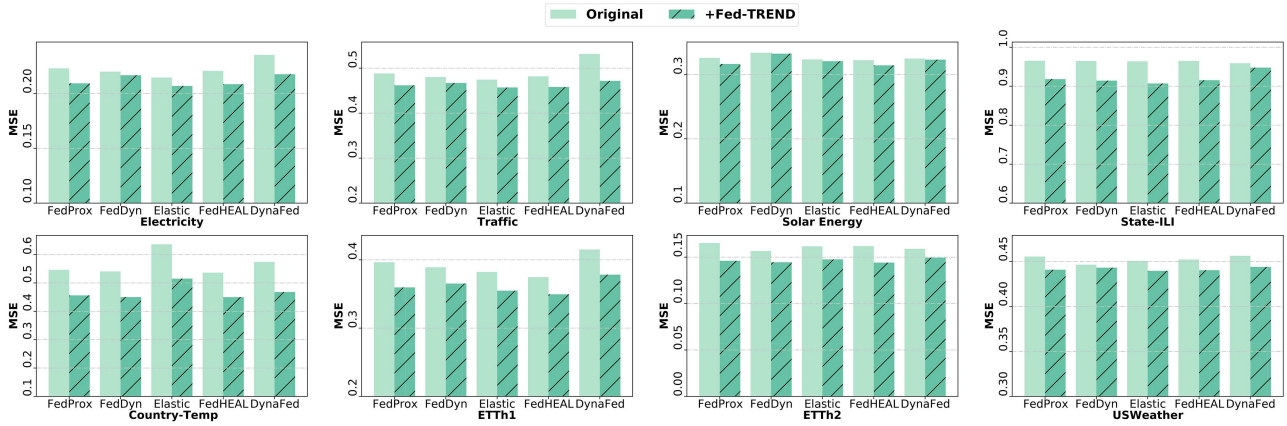

Figure 2 (Color online) The result of using Fed-TREND to improve the general federated learning frameworks.

Table 4 The performance of Fed-TREND with various state-of-the-art time series forecasting models. The best results are in bold.

	Dataset	Electricity		Traffic		Solar energy		State-ILI		Country-Temp		ETTh1		ETTh2		USWeather	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DLinear	FedAvg	0.22199	0.31156	0.48622	0.41120	0.32251	0.43210	0.96516	0.72040	0.54606	0.57863	0.39343	0.42228	0.16318	0.28154	0.45444	0.45011
	Ours	**0.20888**	**0.29838**	**0.46073**	**0.38698**	**0.31457**	**0.42284**	**0.91795**	**0.70034**	**0.45429**	**0.54099**	**0.35814**	**0.39937**	**0.14449**	**0.26381**	**0.44036**	**0.44247**
LightTS	FedAvg	0.22205	0.30894	0.48514	0.40768	0.33548	0.44212	1.24200	0.85050	0.71192	0.67898	0.37634	0.41393	0.16354	0.28388	0.45765	0.45359
	Ours	**0.21006**	**0.29762**	**0.46243**	**0.38758**	**0.33318**	**0.44018**	**1.17520**	**0.82470**	**0.64298**	**0.64576**	**0.35660**	**0.39982**	**0.15203**	**0.27366**	**0.44338**	**0.44723**
TSMixer	FedAvg	0.21727	0.30388	0.47401	0.39677	0.27527	0.35847	1.19635	0.83488	0.54964	0.58744	0.37824	0.41354	0.15104	0.26636	0.44910	0.45031
	Ours	**0.20974**	**0.29598**	**0.46634**	**0.38831**	**0.27441**	**0.35672**	**0.96942**	**0.73354**	**0.51085**	**0.57618**	**0.36611**	**0.40612**	**0.14236**	**0.25674**	**0.44351**	**0.44350**
iTransformer	FedAvg	0.28840	0.38110	0.58663	0.47946	0.27552	0.34224	0.94379	0.72348	0.88088	0.79160	0.49450	0.48204	0.16896	0.29863	0.47024	0.47050
	Ours	**0.28443**	**0.37762**	**0.58295**	**0.47692**	**0.27478**	**0.34109**	**0.87321**	**0.69234**	**0.86617**	**0.78476**	**0.49103**	**0.47991**	**0.16789**	**0.29747**	**0.46755**	**0.46874**

Table 5 The results of the ablation study. "-cu" means it does not construct $\mathcal{D}_{ct}$ that only considers the consistent updated gradients, i.e., using all parameters for $\mathcal{D}_{ct}$ construction. "-$\mathcal{D}_{ct}$" and "-$\mathcal{D}_{gt}$" mean removing the corresponding synthetic datasets. The best results are in bold.

Dataset	Electricity		Traffic		Solar		Iliness		Temperature		ETTh1		ETTh2		USWeather	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Fed-TREND	**0.20888**	**0.29838**	**0.46073**	**0.38698**	**0.31457**	**0.42284**	**0.91795**	**0.70034**	**0.45429**	**0.54099**	**0.35814**	**0.39937**	**0.14449**	**0.26381**	**0.44036**	**0.44247**
-cu	0.21190	0.30099	0.46878	0.39574	0.31428	0.42279	0.91848	0.70060	0.45275	0.53956	0.37340	0.41001	0.15181	0.27123	0.44005	0.44249
-cu, -$\mathcal{D}_{ct}$	0.21524	0.30446	0.47332	0.39979	0.32014	0.42962	0.94931	0.71375	0.47206	0.54721	0.38161	0.41520	0.15537	0.27478	0.44667	0.44655
-$\mathcal{D}_{gt}$	0.21645	0.30591	0.46984	0.39826	0.32307	0.42921	0.95027	0.71408	0.49183	0.55251	0.36022	0.40132	0.14864	0.26832	0.44071	0.44281
-cu, -$\mathcal{D}_{ct}$, -$\mathcal{D}_{gt}$ (FedAvg)	0.22199	0.31156	0.48622	0.41120	0.32251	0.43210	0.96516	0.72040	0.54606	0.57863	0.39343	0.42228	0.16318	0.28154	0.45444	0.45011

5.7 Ablation studies (RQ3)

In Fed-TREND, we introduce the construction of two synthetic datasets: $\mathcal{D}_{ct}$, based on consistent client model updates, and $\mathcal{D}_{gt}$, based on the aggregated global model. The synthetic dataset $\mathcal{D}_{ct}$ is distributed to clients for local training, while $\mathcal{D}_{gt}$ is retained on the central server to finetune and calibrate the global model. In this section, we investigate the effectiveness of $\mathcal{D}_{gt}$, $\mathcal{D}_{ct}$, and the consistent updating ("CU") method used in constructing $\mathcal{D}_{ct}$. The empirical results are displayed in Table 5. When we remove the consistent update dataset construction method, i.e., "-cu", the system's performance slightly declines. This suggests that building $\mathcal{D}_{ct}$ based on consistent updates helps the synthetic data concentrate on capturing more valuable information for model training. Next, we examine the impact of removing the synthetic datasets $\mathcal{D}_{ct}$ ("-CU, -$\mathcal{D}_{ct}$") or $\mathcal{D}_{gt}$ ("-$\mathcal{D}_{gt}$") individually. In both cases, the system's performance significantly decreases across all datasets, demonstrating the importance of each synthetic dataset. Finally, when all synthetic data components are removed, the system degrades to the FedAvg baseline. Overall, the results indicate that each of the proposed components contributes meaningfully to improving model performance.

5.8 Hyperparameter analysis (RQ4)

5.8.1 *The Impact of $\mathcal{D}_{gt}$ dataset size*

Figure 3 illustrates the performance trend as the size of the synthetic dataset $|\mathcal{D}_{gt}|$ increases. Across all datasets, as $|\mathcal{D}_{gt}|$ grows from 0 to 40, model performance improves, but the rate of improvement gradually decreases. Notably, when $|\mathcal{D}_{gt}|$ increases from 0 to 5, system performance improves rapidly, highlighting the positive impact of $\mathcal{D}_{gt}$. However, as $|\mathcal{D}_{gt}|$ continues to grow, the contributions to performance become minimal. This may be because, beyond a certain threshold, additional synthetic data does not provide significant new information.

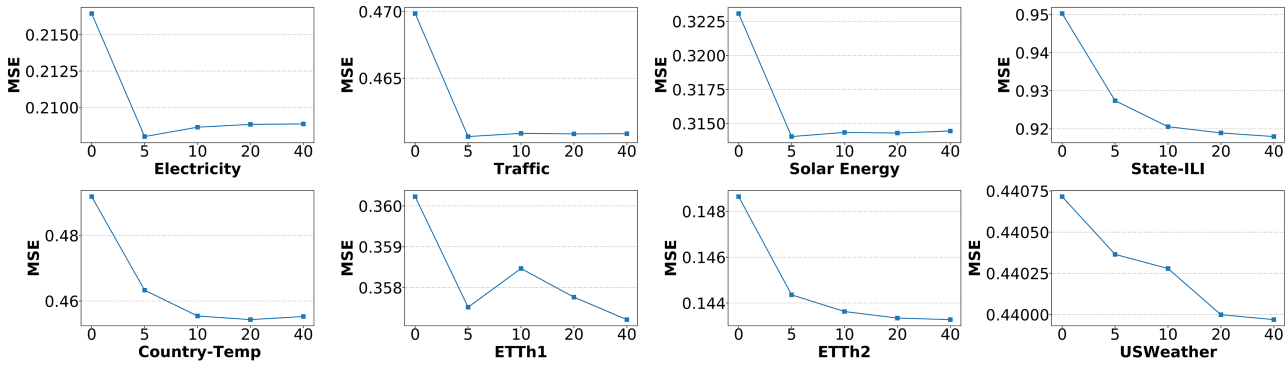

Figure 3 (Color online) The performance trend with different $|\mathcal{D}_{gt}|$.

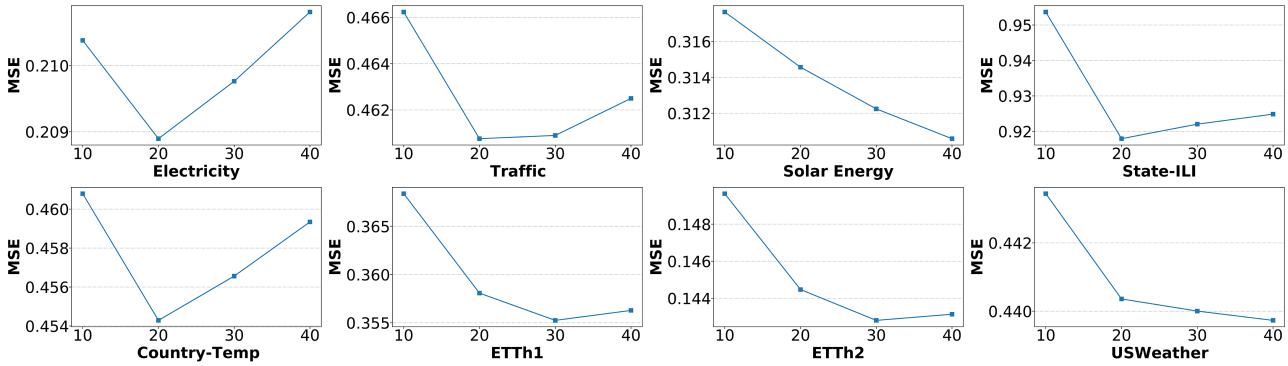

Figure 4 (Color online) The performance trend with different $|\mathcal{D}_{ct}|$.

5.8.2 *The impact of $\mathcal{D}_{ct}$ dataset size*

Figure 4 shows the performance trend of Fed-TREND as the size of $|\mathcal{D}_{ct}|$ increases from 10 to 40. The results indicate that for most datasets, as the size of $\mathcal{D}_{ct}$ grows, model performance initially improves, reaching a peak. This suggests that $\mathcal{D}_{ct}$ provides valuable information for local model training. However, beyond a certain point, further expansion of the dataset size leads to a decline in performance. This phenomenon can be attributed to two main reasons. First, larger synthetic datasets introduce more trainable parameters, increasing the complexity of training and potentially capturing noise. Second, an overly large synthetic dataset may dilute the semantics of clients' original data, ultimately hindering local training. Therefore, selecting an appropriate size for $\mathcal{D}_{ct}$ is crucial for maximizing model performance.

5.8.3 *The impact of input and output time series data length*

To simplify the investigation cases, we assume that input and output data lengths L_x and L_y are the same, and then, we change the data length from the default value 24 to 96. Note that due to the limited number of timesteps in the State-ILI dataset (only 345 in total, as shown in Table 2), we only examine data lengths from 24 to 48 for this dataset. The results are shown in Figure 5.

Intuitively, increasing the data length should decrease model performance since longer forecasting horizons are more challenging. This trend is observed in the Country-Temp, ETTh1, ETTh2, and USWeather datasets. However, in the Electricity, Traffic, and Solar Energy datasets, increasing the data length initially worsens performance, but further increases in data length help mitigate this decline. Interestingly, in the State-ILI dataset, a longer data length improves model forecasting. This may be because, while a longer output data length increases forecasting difficulty, a longer input data length provides more valuable contextual information. For example, in real-world scenarios, illness statistics like those in the State-ILI dataset exhibit seasonality, so having a longer observed data window is beneficial for accurate predictions.

Overall, in all cases, Fed-TREND consistently outperforms its corresponding base federated learning framework, FedAvg, by a significant margin, demonstrating the robustness and effectiveness of our method across different data length settings.

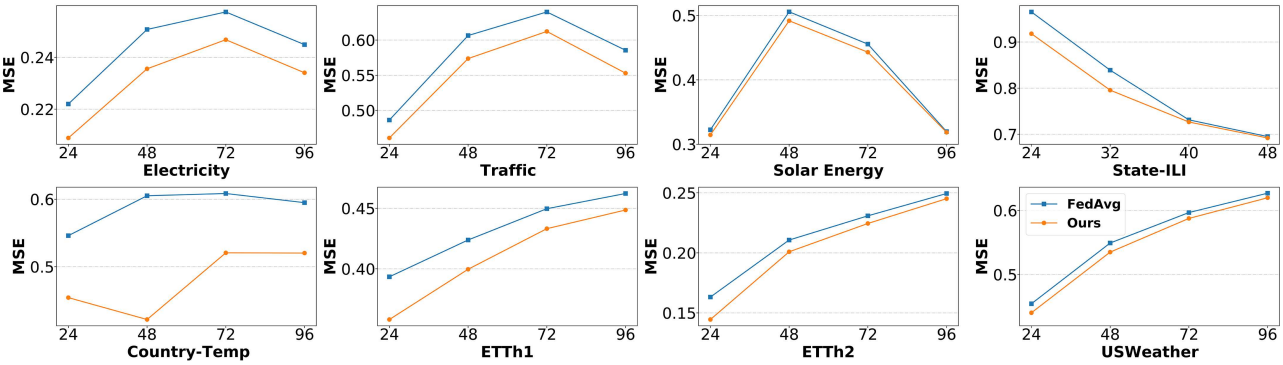

Figure 5 (Color online) The performance trend with time series data length.

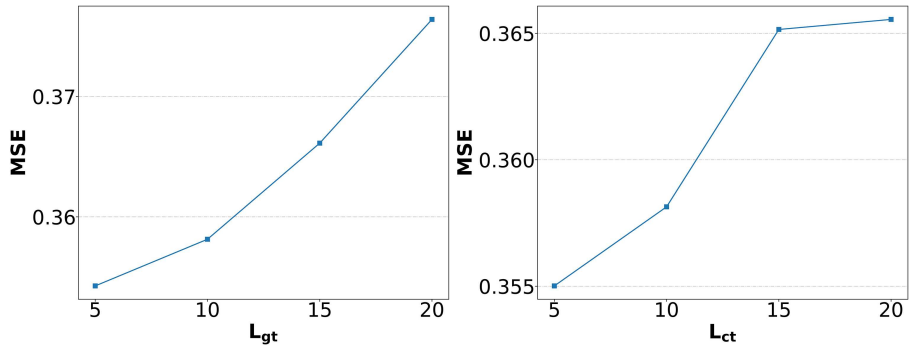

Figure 6 (Color online) The performance trend with the synthetic data construction frequency L_{gt} and L_{ct} on ETTh1. A similar trend can be observed on other datasets.

5.8.4 *The impact of data construction interval*

Figure 6 illustrates the impact of synthetic data construction frequency on system performance. Due to space limitations, we present the results for the ETTh1 dataset, but similar trends are observed across the other seven datasets. The results indicate that smaller values of L_{gt} and L_{ct} lead to better performance. This is because more frequent construction of synthetic datasets allows them to quickly adapt to recent dynamics, thereby incorporating the latest knowledge from model updates. However, frequently updating synthetic datasets also increases computational costs, creating a trade-off between effectiveness and efficiency. In our experiments, we found that setting L_{gt} and L_{ct} to 10 strikes a balance.

5.9 Further study with privacy protection mechanism (RQ5)

To enhance privacy protection, federated learning often incorporates privacy mechanisms. Among these, local differential privacy (LDP) is considered the gold standard and the most widely used approach [59]. In this section, we evaluate whether Fed-TREND can still improve the performance of baseline federated learning when using LDP. Specifically, we implement LDP with the Laplace mechanism by adding noise sampled from $\mathcal{N}(0, \lambda^2\mathbf{I})$ to the model parameters [60], where $\mathcal{N}$ represents the normal distribution, and we set $\lambda = 0.001$ to balance the trade-off between performance and privacy. As shown in Figure 7, integrating Fed-TREND with "FedAvg + LDP" results in lower MSE scores across all datasets, indicating that Fed-TREND remains effective in the context of differential privacy.

6 Conclusion

This paper introduces Fed-TREND, a federated time series forecasting framework designed to close the performance gap between federated and centralized time series forecasting by enhancing learning on heterogeneous data. Specifically, Fed-TREND constructs two types of synthetic datasets based on clients' uploaded models and the aggregated global model to improve the consensus during clients' local training and to refine the global model aggregation, respectively. Since the synthetic data construction process does not require any prior knowledge and is performed on the central server, Fed-TREND can be easily integrated with most federated learning frameworks without imposing a heavy computational burden on clients. Extensive experiments conducted on eight time series datasets

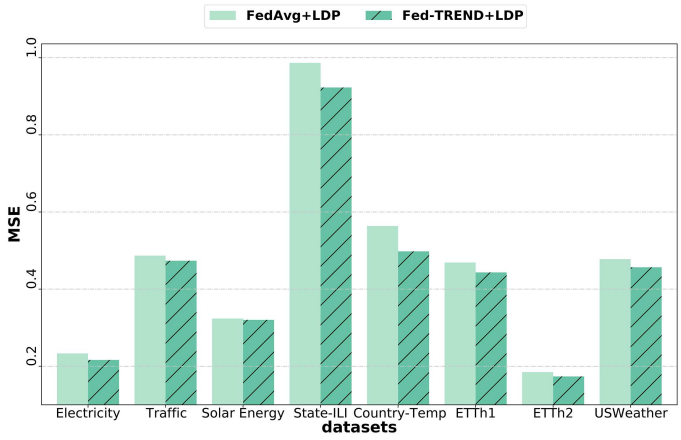

Figure 7 (Color online) The performance comparison of the base federated learning and the one equipping of Fed-TREND under the context of differential privacy.

using four popular forecasting models demonstrate the effectiveness and generalization capabilities of the proposed Fed-TREND.

Limitation and future work. As an initial attempt to use synthetic data for addressing data heterogeneity in federated time series forecasting, this work presents several areas for further improvement. First, since part of the synthetic data is generated based on clients' uploaded model updates, it may be sensitive to noise and abnormal updates. This limitation constrains the noise scale that can be used when applying differential privacy and may leave the system more vulnerable to adversarial attacks. Additionally, to reduce the computational burden on clients, we shift the entire synthetic data construction process to the central server. Although the central server typically has significant computational resources, as the dataset size and number of clients increase, the overhead of constructing synthetic data may become a bottleneck. Future work can explore more robust and computationally efficient frameworks to address these challenges.

Acknowledgements This work was supported by Australian Research Council under the streams of Future Fellowship (Grant No. FT210100-624), Discovery Project (Grant No. DP240101108), and Linkage Project (Grant No. LP230200892).

Open Access funding enabled and organized by CAUL and its Member Institutions.

References

1 Cai R C, Wu Y J, Huang X K, et al. Granger causal representation learning for groups of time series. Sci China Inf Sci, 2024, 67: 152103
2 Sun L, Wang Y Y, Ren Y J, et al. Path signature-based XAI-enabled network time series classification. Sci China Inf Sci, 2024, 67: 170305
3 Li Y, Lu X, Xiong H, et al. Towards long-term time-series forecasting: feature, pattern, and distribution. In: Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE), 2023. 1611–1624
4 Zhou H, Zhang S, Peng J, et al. Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 11106–11115
5 Liu Y, Hu T, Zhang H, et al. itransformer: inverted transformers are effective for time series forecasting. In: Proceedings of the Twelfth International Conference on Learning Representations, 2024
6 Vaswani A. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, 2017
7 Zhang T, Zhang Y, Cao W, et al. Less is more: fast multivariate time series forecasting with light sampling-oriented MLP structures. ArXiv:2207.01186
8 Zeng A, Chen M, Zhang L, et al. Are transformers effective for time series forecasting? In: Proceedings of AAAI Conference on Artificial Intelligence, 2023. 11121–11128
9 Chen S A, Li C L, Arik S O, et al. TSMixer: an All-MLP architecture for time series forecast-ing. Trans Mach Learn Res, 2023. https://openreview.net/forum?id=wbpxTuXgm0
10 Asghar M R, Dán G, Miorandi D, et al. Smart meter data privacy: a survey. IEEE Commun Surv Tutor, 2017, 19: 2820–2835
11 McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of Artificial Intelligence and Statistics, 2017. 1273–1282
12 Li Q, Diao Y, Chen Q, et al. Federated learning on non-IID data silos: an experimental study. In: Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), 2022. 965–978
13 Ye M, Fang X, Du B, et al. Heterogeneous federated learning: state-of-the-art and research challenges. ACM Comput Surv, 2023, 56: 1–44
14 Deng J, Chen X, Jiang R, et al. A multi-view multi-task learning framework for multi-variate time series forecasting. IEEE Trans Knowl Data Eng, 2022, 35: 7665–7680

15 Jin M, Zheng Y, Li Y F, et al. Multivariate time series forecasting with dynamic graph neural ODEs. IEEE Trans Knowl Data Eng, 2022, 35: 9168–9180
16 Cazenavette G, Wang T, Torralba A, et al. Dataset distillation by matching training trajectories. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022. 4750–4759
17 Yu R, Liu S, Wang X. Dataset distillation: a comprehensive review. IEEE Trans Pattern Anal Mach Intell, 2024, 46: 150–170
18 Gao X, Yu J, Jiang W, et al. Graph condensation: a survey. ArXiv:2401.11720
19 Goetz J, Tewari A. Federated learning via synthetic data. ArXiv:2008.04489
20 Xiong Y, Wang R, Cheng M, et al. Feddm: iterative distribution matching for communication-efficient federated learning. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 16323–16332
21 Liu P, Yu X, Zhou J T. Meta knowledge condensation for federated learning. In: Proceedings of the Eleventh International Conference on Learning Representations, 2023
22 Wang Y, Fu H, Kanagavelu R, et al. An aggregation-free federated learning for tackling data heterogeneity. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024. 26233–26242
23 Benidis K, Rangapuram S S, Flunkert V, et al. Deep learning for time series forecasting: tutorial and literature survey. ACM Comput Surv, 2022, 55: 1–36
24 Ma Q, Liu Z, Zheng Z, et al. A survey on time-series pre-trained models. IEEE Trans Knowl Data Eng, 2024, 36: 7536–7555
25 Zhang M, Ding D, Pan X, et al. Enhancing time series predictors with generalized extreme value loss. IEEE Trans Knowl Data Eng, 2021, 35: 1473–1487
26 Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Phys D, 2020, 404: 132306
27 Liu F, Zhou X, Cao J, et al. Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional LSTM-CNN. IEEE Trans Knowl Data Eng, 2020, 34: 2626–2640
28 LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE, 2002, 86: 2278–2324
29 Lai G, Chang W C, Yang Y, et al. Modeling long-and short-term temporal patterns with deep neural networks. In: Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018. 95–104
30 Shih S Y, Sun F K, Lee H. Temporal pattern attention for multivariate time series forecasting. Mach Learn, 2019, 108: 1421–1441
31 Wen Q, Zhou T, Zhang C, et al. Transformers in time series: a survey. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, 2023. 6778–6786
32 Yang Q, Liu Y, Chen T, et al. Federated machine learning: concept and applications. ACM Trans Intell Syst Tech, 2019, 10: 1–19
33 Collins L, Hassani H, Mokhtari A, et al. Exploiting shared representations for personalized federated learning. In: Proceedings of International Conference on Machine Learning, 2021. 2089–2099
34 Lin T, Kong L, Stich S U, et al. Ensemble distillation for robust model fusion in federated learning. Adv Neural Inform Process Syst, 2020, 33: 2351–2363
35 Chen H Y, Chao W L. Fedbe: Making Bayesian model ensemble applicable to federated learning. In: Proceedings of International Conference on Learning Representations, 2020
36 Liu Q, Chen C, Qin J, et al. Feddg: federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021. 1013–1023
37 Yoon T, Shin S, Hwang S J, et al. Fedmix: approximation of mixup under mean augmented federated learning. In: Proceedings of International Conference on Learning Representations, 2021
38 Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks. Proc Mach Learn Syst, 2020, 2: 429–450
39 Karimireddy S P, Kale S, Mohri M, et al. Scaffold: Stochastic controlled averaging for federated learning. In: Proceedings of International Conference on Machine Learning, 2020. 5132–5143
40 Acar D A E, Zhao Y, Navarro R M, et al. Federated learning based on dynamic regularization. ArXiv:2111.04263
41 Chen D, Hu J, Tan V J, et al. Elastic aggregation for federated optimization. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 12187–12197
42 Chen Y, Huang W, Ye M. Fair federated learning under domain skew with local consistency and domain diversity. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024. 12077–12086
43 Zhou Y, Pu G, Ma X, et al. Distilled one-shot federated learning. ArXiv:2009.07999
44 Hu S, Goetz J, Malik K, et al. Fedsynth: gradient compression via synthetic data in federated learning. In: Proceedings of Workshop on Federated Learning: Recent Advances and New Challenges, 2022
45 Zhang J, Chen C, Li B, et al. Dense: Data-free one-shot federated learning. Adva Neural Inform Process Syst, 2022, 35: 21414–21428
46 Dai R, Zhang Y, Li A, et al. Enhancing one-shot federated learning through data and ensemble co-boosting. In: Proceedings of the Twelfth International Conference on Learning Representations, 2024
47 Pi R, Zhang W, Xie Y, et al. Dynafed: tackling client data heterogeneity with global dynamics. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 12177–12186
48 Liu Q, Liu X, Liu C, et al. Time-ffm: towards LM-empowered federated foundation model for time series forecasting. ArXiv:2405.14252
49 Abdel-Sater R, Hamza A B. A federated large language model for long-term time series forecasting. ArXiv:2407.20503
50 Yan Y, Yang G, Gao Y, et al. Multi-participant vertical federated learning based time series prediction. In: Proceedings of the 8th International Conference on Computing and Artificial Intelligence, 2022. 165–171
51 Zhao B, Mopuri K R, Bilen H. Dataset condensation with gradient matching. ArXiv:2006.05929
52 Li G, Togo R, Ogawa T, et al. Dataset distillation using parameter pruning. IEICE Transactions on Fundamentals of Electronics. Commun Comput Sci, 2024, 107: 936–940
53 Du J, Jiang Y, Tan V Y, et al. Minimizing the accumulated trajectory error to improve dataset distillation. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 3749–3758
54 Zhang C, Xie Y, Bai H, et al. A survey on federated learning. Knowledge-Based Syst, 2021, 216: 106775
55 Dwork C. Differential privacy. In: Proceedings of International Colloquium on Automata, Languages, and Programming, 2006. 1–12
56 Yin X, Zhu Y, Hu J. A comprehensive survey of privacy-preserving federated learning. ACM Comput Surv, 2021, 54: 1–36
57 Sutskever I, Martens J, Dahl G, et al. On the importance of initialization and momentum in deep learning. In: Proceedings of International Conference on Machine Learning, 2013. 1139–1147
58 Kingma D P. Adam: a method for stochastic optimization. ArXiv:1412.6980
59 Zhang S, Yuan W, Yin H. Comprehensive privacy analysis on federated recommender system against attribute inference attacks. In: Proceedings of IEEE Transactions on Knowledge and Data Engineering, 2023
60 Yang M, Guo T, Zhu T, et al. Local differential privacy and its applications: a comprehensive survey. Comput Stand Inter, 2023, 89: 103827