Tackling Data Heterogeneity in Federated Time Series Forecasting

Wei Yuan
The University of Queensland
Brisbane, Australia
w.yuan@uq.edu.au

Guanhua Ye
Beijing University of Posts and Telecommunications
Beijing, China
g.ye@bupt.edu.cn

Xiangyu Zhao
City University of Hong Kong
Hongkong, China
xianzhao@cityu.edu.hk

Quoc Viet Hung Nguyen
Griffith University
Gold Coast, Australia
henry.nguyen@griffith.edu.au

Yang Cao
Institute of Science Tokyo
Tokyo, Japan
cao@c.titech.ac.jp

Hongzhi Yin
The University of Queensland
Brisbane, Australia
h.yin1@uq.edu.au

***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 are inherently heterogeneous. In this paper, we propose a novel framework, 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.**

I. Introduction

With the proliferation of sensors, wearables, and Internet of Things (IoT) devices, the volume of time series data has increased dramatically in recent years. Time series forecasting has emerged as a focal point for both academic and industrial communities, reflecting its importance of 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 [1]. For example, several studies [2], [3] have refined the Transformer architecture [4] to efficiently handle long-sequence time series predictions. Meanwhile, recent research [5]–[7] 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 [8]. Similarly, smart wearables record detailed personal health metrics, such as oxygen saturation, heart rate, ECG, and EEG. Given the sensitive nature of this data, sharing it significantly heightens the risk of data breaches.

Federated learning [9], [10] 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 severing various applications.

[1]https://gdpr-info.eu/
[2]https://oag.ca.gov/privacy/ccpa

Existing federated learning methods face significant challenges in effectively learning from such heterogeneous data [11].

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 [7]. 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 produce a suboptimal global model. Addressing these heterogeneity challenges in cross-device federated time series forecasting remains a largely unexplored research area.

To address the challenges of learning from heterogeneous time series data, we propose a novel method Fed-TREND (Federated Time Series Forecasting with Synthetic Data). Fed-TREND draws inspiration from recent advancements in data condensation [12]–[14], where synthetic data is generated from model trajectories to encapsulate the essential information. Specifically, Fed-TREND generates two types of synthetic data on the central server to tackle the two forms of heterogeneity respectively, enhancing the federated learning process. The first type of synthetic data, $\mathcal{D}_{ct}$, is generated based on model updates uploaded by all clients, encapsulating the distribution 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 challenges posed by heterogeneous temporal patterns.

Noted that unlike other data condensation-based federated learning approaches [15]–[18], where synthetic data acts as the primary information carrier for collaborative learning, Fed-TREND uses synthetic data solely 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 often difficult to guarantee. Furthermore, the synthetic data construction occurs entirely on the central server, minimizing the computational burden on client devices.

To sum up, 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 [6], LightTS [5], TSMixer [7], and iTransformer [3]. The results consistently demonstrate that Fed-TREND significantly improves the federated forecasting performance across all time series forecasting datasets from diverse application scenarios.

The remainder of this work is organized as follows. In Section II, we present a literature review of related topics, followed by introducing the preliminaries of time series forecasting in Section III. Section IV describes the technical details of our Fed-TREND. Extensive empirical results and analysis are shown in Section V. Finally, a brief summarization of this paper is delivered in Section VI.

II. Related Work

In this section, we briefly review the literature of four related topics: time series forecasting, federated learning with data heterogeneity, federated learning with data condensation, and federated time series forecasting.

A. Time Series Forecasting

Time series forecasting is a widely used task applicable to numerous real-world scenarios, including energy consumption prediction [19], pandemic spread modeling [20], weather and traffic forecasting [21], and more [22]. Early studies in time series forecasting relied on statistical methods, such as autoregressive integrated moving average (ARIMA) [23], exponential smoothing [24], and structural models [25]. However, these methods require extensive expert effort to develop.

In recent years, deep learning-based approaches have become the dominant trend in time series forecasting [22]. These methods typically use neural network architectures, such as RNNs [26], CNNs [27], Transformers [4], and MLPs, as backbone models. For example, LSTNet [28] and TPR-LSTM [29] combine CNNs and RNNs with attention mechanisms to capture both short- and long-term dependencies in time series data. However, RNN-based methods often suffer from issues

like gradient explosion or vanishing, while CNNs are limited in modeling long-term sequences due to the restricted size of their receptive fields. Transformers have recently gained popularity in time series forecasting due to their ability to model global dependencies [30]. Zhou et al. [2] introduced Informer, which reduces time complexity and enhances memory efficiency. Autoformer [31] introduces a decomposition architecture with an auto-correlation mechanism, and Liu et al. [3] innovatively reverse the data dimensions in the Transformer's attention and feed-forward layers, achieving improved performance. The use of pure MLPs in time series forecasting has also become a recent trend because of their simplicity of implementation and effectiveness of performance. Zeng et al. [6] challenged the effectiveness of Transformers in time series forecasting domain, proposing a MLP-based model called DLinear. LightTS [5] incorporates two down-sampling methods, interval sampling and continuous sampling, to enhance MLP performance, while TSMixer [7] uses mixing operations across both time and feature dimensions to efficiently capture relevant information.

However, all of these methods are implemented in a centralized manner, overlooking the practical challenges of data privacy in real-world applications.

B. Federated Learning with Data Heterogeneity

Federated learning enables collaborative training of a global model without requiring access to clients' raw data and has been widely researched in many domains [32]–[36]. This learning paradigm has garnered significant attention for applications where data collection is challenging [37], [38]. FedAvg [9] 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 is heterogeneous, a common situation as clients independently collect data in diverse environments.

To address data heterogeneity in federated learning, numerous methods have been proposed [10], [11]. 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 [39]–[41] or that data transmission between clients is allowed [42], [43]. 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 [44] introduces a proximal term during local model training to prevent local updates from deviating too far from the global model. SCAFFOLD [45] employs a control variate for variance reduction to stabilize aggregation. FedDyn [46] uses dynamic regularization to adjust each client's objective during training. Elastic [47] designs an elastic aggregation approach that dampens the influence of updates to sensitive parameters. Chen et al. [48] propose 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.

C. Federated Learning with Data Condensation

Data condensation aims to compress a large training dataset into a smaller, synthetic dataset [13] and has recently been integrated into federated learning [14]. This integration serves two primary purposes: (1) to improve communication efficiency [49]–[53] and (2) to address data heterogeneity [15]–[18], [54], [55]. Here, we focus on the latter. Goetz et al. [15] and Xiong et al. [16] propose 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. [18] extend 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. DynaFed [54] is more closely related to our approach, FedTREND, but it only uses synthetic data to adjust the global model and is designed for image classification tasks.

D. Federated Learning in Time Series Forecasting

The research of federated time series forecasting is still under-explored. Time-FFM [56] 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 et al. [57] apply organization-level federated learning to train a time series forecasting model based on large language models. Yan et al. [58] propose 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.

III. Preliminaries

In this section, we present a formal introduction for the settings of federated time series forecasting and then briefly introduce the basic time series forecasting models. Note that,

TABLE I
LIST OF IMPORTANT NOTATIONS.

c_i	a client/device in federated time series forecasting
$\mathcal{D}_{c_i}$	the local dataset for client c_i.
$\mathcal{D}_{ct}$	synthetic dataset generated using clients' model trajectories.
$\mathcal{D}_{gt}$	synthetic dataset generated using global model trajectories.
$\mathcal{T}_{ct}$	trajectories bank for client model updates.
$\mathcal{T}_{gt}$	trajectories bank for 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 $\mathcal{D}_{gt}$
$\mathbf{W}_{c_i}^t$	the model trained by c_i at round t
$\mathbf{W}^t$	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

we use squiggle uppercase (e.g., $\mathcal{A}$) to indicate set or algorithms, bold lowercase (e.g., $\mathbf{a}$) to represent vectors, and bold uppercase (e.g., $\mathbf{A}$) to denote matrices or tensors. Table I lists some important notations.

A. 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 $\mathbf{X}_{c_i} = [\mathbf{x}_1^{c_i}, \mathbf{x}_2^{c_i}, \ldots, \mathbf{x}_{T-1}^{c_i}, \mathbf{x}_T^{c_i}] \in \mathbb{R}^{T \times f}$, where T is the total lengths of the data and f is the number of dimensions. Notebly, in federated time series forecasting, the time series data $\mathbf{X}_{c_i}$ are always kept on corresponding device and will 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:

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

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

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

Federated time series forecasting employs a central server to coordinate clients to optimize E.q. 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 a 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 E.q. 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 received the updated parameters, the central server aggregates these parameters to form a new global model parameters:

$$\mathbf{W}^{t+1} \leftarrow agg(\{\mathbf{W}_{c_i}^t\}_{c_i \in \mathcal{C}^t}) \tag{3}$$

One main-stream aggregation solution is FedAvg [9], which utilizes weighted average to aggregate client uploaded parameters:

$$\mathbf{W}^{t+1} = \sum_{c_i \in \mathcal{C}^t} \frac{|\mathcal{D}_{c_i}|}{\sum_{c_j \in \mathcal{C}^t} |\mathcal{D}_{c_j}|} \mathbf{W}_{c_i}^t \tag{4}$$

This design performs well when the client data are homogeneous. However, when client data are heterogeneous, local models are updated towards local optimal solution and FedAvg cannot simply aggregate them to achieve optimal global performance.

B. 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 and MLP.

DLinear [6]: DLinear decomposes input time series data into seasonal and trend components using a moving average kernel. For each component, DLinear employs a linear layer network, summing the resulting features for prediction.

LightTS [5]: LightTS utilizes two down-sampling strategies, continuous sampling and interval sampling, to process time series data. Besides, it introduces an Information Exchange Block (IEBblock), which consists of two MLPs that encode input matrix data from both the row and column perspectives.

TSMixer [7]: Generally, TSMixer consists of four main components: a time-mixing MLP for capturing temporal patterns, a feature-mixing MLP for leveraging covariate information across time steps, a temporal projection layer to adjust the input length for forecasting, and residual connections that link the MLPs to enhance model depth.

iTransformer [3]: Unlike other Transformer-based forecasting models [2], iTransformer retains the original Transformer architecture but inverts the input dimensions. It encodes the full history of each variable into a single token, using the attention mechanism to capture correlations between variables instead of time steps. This approach inherently encodes temporal information, making positional encoding unnecessary.

IV. METHODOLOGY

In this section, we firstly provide the overview and motivation of developing Fed-TREND. After that, we detailly describe the techniques of each component in Fed-TREND.

A. Overview of Fed-TREND

In Section I, 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

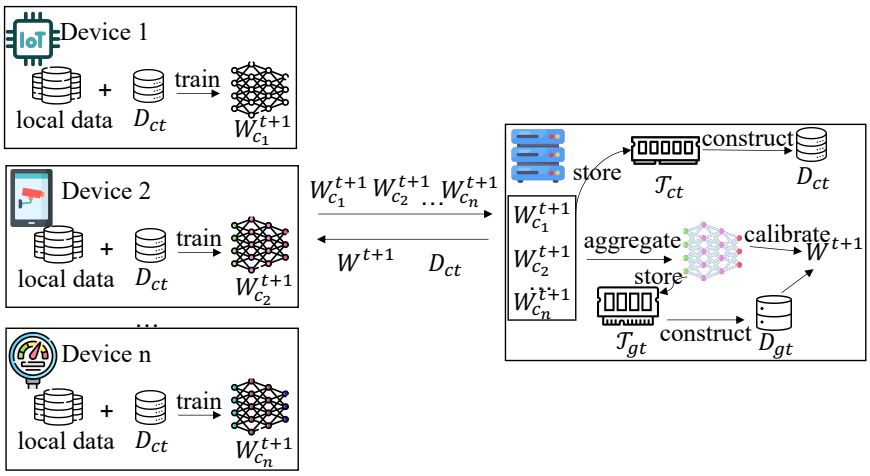

Fig. 1. The overall architecture of Fed-TREND. When clients uploaded their model updates, these updates are (1) used for aggregation as the original federated learning and (2) stored in a trajectory bank $\mathcal{T}_{ct}$ for $\mathcal{D}_{ct}$ construction. In addition, the aggregated global model 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.

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 are corresponding 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 in 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 Section III-A. In the experimental section part, we will also evaluate the empirical performance of Fed-TREND when integrated with various federated learning frameworks.

B. Synthetic Data $\mathcal{D}_{ct}$ for Representative Knowledge Transfer

The recently developed technique of data condensation [12] 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 updates 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:

$$\underset{\mathcal{D}_{ct}}{argmin}\, \mathbb{E}_{c_j \sim U(\mathcal{C}), s \sim U(1, t-L_{ct})} \left[d(\widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}}, \mathbf{W}_{c_j}^{s+L_{ct}}) \right] \quad (5)$$

$$s.t. \widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}} = argmin\, \mathcal{L}(\mathbf{W}_{c_j}^{s}, \mathcal{D}_{ct}) \quad (6)$$

Here, L_{ct} is the segment length of a trajectory and $U(\cdot)$ denotes uniform random sampling. The meaning of the above two formulas is that, in E.q. 6, 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_c steps, obtaining the trained model $\widetilde{\mathbf{W}}_{c_j}^{s+L_{ct}}$. Note that same as the settings in real dataset, $\mathbf{X}_i^{ct}$ and $\mathbf{Y}_i^{ct}$ are the input and target output pair. Then, in E.q. 5, 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 $(\mathbf{X}_i^{ct}, \mathbf{Y}_i^{ct})$ are learnable parameters.

Intuitively, by optimizing Eqs. 5 and 6, 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 when 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 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, E.q. 5 is simplied to:

$$\underset{\mathcal{D}_{ct}}{argmin}\, \mathbb{E}_{c_j \sim U(\mathcal{C})} \left[d(\widetilde{\mathbf{W}}_{c_j}^{k*L_{ct}}, \mathbf{W}_{c_j}^{(k+1)*L_{ct}}) \right] \tag{7}$$

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 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 [48], the parameters that consistently update towards a direction may reflect some important signals and learn from these parameters can make the synthetic dataset more concentrate on extracting these knowledge. Therefore, before the parameters $\widetilde{\mathbf{W}}_{c_j}^{k*L_{ct}}$ been added to the trajectories memory bank $\mathcal{T}_{ct}$, the central server will firstly 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., $sign(\Delta w_{c_j,m}^{k*L_{ct}-1}) == sign(\Delta w_{c_j,m}^{k*L_{ct}})$. If the update is not consistent, the distance loss of corresponding element in E.q. 7 will be masked.

After the central server constructed the synthetic dataset $\mathcal{D}_{ct}$, it will be dispersed to each client and mixed with 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.

C. Synthetic Data for Global Model Refinement

Although the synthetic data $\mathcal{D}_{ct}$ can carry some representative information from all clients to improve the local training consensus, the heterogeneity problem still persists. This is because that the size of $\mathcal{D}_{ct}$ is limited considering the communication cost and it is hard to let $\mathcal{D}_{ct}$ capture all information of clients' data. Consequently, the global model may still drift away from the optimal point after aggregation. 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 aggregated global model $\mathbf{W}^t$ at each

Algorithm 1 The pseudo-code for 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 $\mathbf{W}^0$
$\mathcal{T}_{ct} = \{c_i : [\mathbf{W}_{c_i}^0]\}_{c_i \in \mathcal{C}}$, $\mathcal{T}_{gt} = \{\mathbf{W}^0\}$
$\mathcal{D}_{ct} = \emptyset, \mathcal{D}_{gt} = \emptyset$
for each round $t = 0, ..., R-1$ **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
    $\mathbf{W}_{c_i}^{t+1} \leftarrow$ CLIENTTRAIN(c_i, $\mathbf{C}^t$, $\mathcal{D}_{ct}$)
  end for
  // execute on central server
  if $t\%L_{ct} == 0$ **then**
    check each element's update direction consistency
    append $\mathbf{W}_{c_i}^{t+1}$ into $\mathcal{T}_{ct}$
  end if
  $\mathbf{W}^{t+1} \leftarrow$ aggregate received client model parameters $\{\mathbf{W}_{c_i}^{t+1}\}_{c_i \in \mathcal{C}^t}$
  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**
  $\mathcal{D}_{ct} \leftarrow$ DATACONSTRUCTION($\mathcal{T}_{ct}$)
end if
if $t\%L_{gt} == 0$ **then**
  $\mathcal{D}_{gt} \leftarrow$ DATACONSTRUCTION($\mathcal{T}_{gt}$)
end if
function CLIENTTRAIN(c_i, $\mathbf{W}^t$, $\mathcal{D}_{ct}$)
  download $\mathbf{W}^t$ and $\mathcal{D}_{ct}$
  $\mathbf{W}_{c_i}^{t+1} \leftarrow$ update local model with forecasting objective E.q. 2 on $\mathcal{D}_{c_i}$ and $\mathcal{D}_{ct}$
  return $\mathbf{W}_{c_i}^{t+1}$
end function
function DATACONSTRUCTION($\mathcal{T}$)
  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
  return $\mathcal{D}_{syn}$
end function

round. Then, $\mathcal{D}_{gt}$ is optimized as follows:

$$\underset{\mathcal{D}_{gt}}{argmin}\, \mathbb{E}_{s\sim U(1,t-L_{gt}^{seg})}\left[d(\widetilde{\mathbf{W}}^{s+L_{gt}^{seg}}, \mathbf{W}^{s+L_{gt}^{seg}})\right] \quad (8)$$

$$s.t. \widetilde{\mathbf{W}}^{s+L_{gt}^{seg}} = argmin\, \mathcal{L}(\mathbf{W}^s, \mathcal{D}_{gt}) \quad (9)$$

where L_{gt}^{seg} is the length of the trajectory segment and $\widetilde{\mathbf{W}}^{s+L_{gt}^{seg}}$ is the model trained on $\mathcal{D}_{gt}$ from the initialization point of $\widetilde{W}^s$ for L_{gt}^{seg} steps. Similar to $\mathcal{D}_{ct}$, $\mathcal{D}_{gt}$ is constructed with pairs of learnable input and target outputs $\mathcal{D}_{gt} = \{(\mathbf{X}_i^{gt}, \mathbf{Y}_i^{gt})\}_{i=1}^{|\mathcal{D}_{gt}|}$ and has been optimized using E.q. 8. 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.

D. Implementation of Synthetic Data Construction

We employ the most commonly used synthetic data construction algorithm MTT [12] 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 [59], PP [60], FTD [61], and so on. The detailed steps for synthetic data construction are outlined in Algorithm 1, Lines 29-36. The integration of $\mathcal{D}_{ct}$ and $\mathcal{D}_{gt}$ into the general federated learning framework is described in Lines 11-23 and Line 26 in Algorithm 1.

E. Discussion

In this part, we briefly discuss Fed-TREND from three aspects: privacy, communication and computational burden.

1) Privacy Analysis: The synthetic data in Fed-TREND are constructed in accordance with the standard federated learning protocol without any additional assumptions. Hence, the privacy-preserving capabilities of Fed-TREND should be consistent with those of traditional federated learning methods. In Section V-I, we also showcase the compatibility of Fed-TREND with existing privacy-preserving mechanisms [62].

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 $8\times20\times(size(\mathbf{X}^{ct})+size(\mathbf{Y}^{ct}))$. Given that $size(\mathbf{X}^{ct})$ is a sequence of 24 numbers in our case, the total additional cost is less than 30KB per client.

3) Computational Burden Analysis: Existing condensation techniques often suffer from high computational costs. To mitigate this issue in federated learning systems, unlike other works [15]–[18] 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.

V. 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?
- **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 privacy protection mechanism?

A. 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 II. ETTh [3] datasets record hourly electrical transformer statistics over a span of two years. Electricity [4] captures the electricity consumption in kilowatts of 321 clients from 2012 to 2014. Traffic [5] tracks the hourly road occupancy rate using 862 sensors across the San Francisco Bay Area freeways over 48 months (2015-2016). Solar Energy [6] provides solar power production records at 10-minute intervals in 2006 from 137 PV plants in Alabama State. State-ILI [7] tracks Influenza-like Illness (ILI) data weekly for 37 U.S. states from approximately 2009 to 2017. Country-Temp [8] contains the average land temperature of 131 countries from 1823 to 2013. USWeather [9] includes 4 years from 2010 to 2013 climatological data and we follow the usage in [2]. For each dataset, 70% of the data is used for training and validation, while the remaining 30% is reserved for testing. In this paper, we focus on cross-device federated time series forecasting, where each device (e.g., sensors, meters, wearables, etc) is treated as a client. Consequently, in ETTh1, ETTh2, and USWeather, clients track different variables, whereas in

[3]https://github.com/zhouhaoyi/ETDataset
[4]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[5]http://pems.dot.ca.gov
[6]http://www.nrel.gov/grid/solar-power-data.html
[7]https://github.com/emilylaiken/ml-flu-prediction
[8]https://data.world/data-society/global-climate-change-data
[9]https://www.ncei.noaa.gov/data/local-climatological-data/

TABLE II
THE STATISTICS OF DATASETS.

Dataset	Client Num	Timesteps	Granularity	Domain	Start Time
ETTh1, ETTh2	7	14,400	1 hour	Energy	2016/7/1
Electricity	321	26,304	1 hour	Energy	2012/1/1
Traffic	862	17,544	1 hour	Traffic	2015/1/1
Solar Energy	137	52,560	10 minutes	Energy	2006/1/1
State-ILI	37	345	1 week	Disease	2009/10/4
Country-Temp	131	2,277	1 month	Climate	1823/1/1
USWeather	12	35,064	1 hour	Climate	2010/1/1

the remaining datasets, clients record the same variable for different entities.

B. Evaluation Metrics

Following previous time series forecasting studies [3], [5]–[7], 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. Lower MSE and MAE values indicate better forecasting accuracy.

C. Baselines

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

- **Centralized**: This traditional approach trains a time series forecasting model by collecting data from all devices and training it on a central server.
- **FedAvg** [9]: The most widely used federated learning framework, FedAvg averages the model parameters uploaded by clients to update the global model.
- **FedProx** [44]: An extension of FedAvg, FedProx introduces a proximal term in the local training objective to stabilize updates from clients with diverse data distributions.
- **FedDyn** [46]: 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** [47]: 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** [48]: A state-of-the-art technique designed to address domain skew, FedHEAL maintains both local consistency and domain diversity, enhancing the global model's generalization across different client domains.
- **DynaFed** [54]: Similar to our $\mathcal{D}_{gc}$ construction process, DynaFed is designed for image classification and does not update the synthetic data during training. Consequently, its performance degrades as the synthetic data becomes outdated over time.

D. Implementation Details

For the main experiments, we use DLinear [6] as the default base model for time series forecasting in both Fed-TREND and the baselines, considering its effectiveness and efficiency. In Section V-F, 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 will examine the impact of these lengths in Section V-H3. The total number of global training 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 [63] 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 are explored in Section V-H4. The sizes of $\mathcal{D}_{gt}$ and $\mathcal{D}_{ct}$ are explored in Section V-H2 and Section V-H1, respecatively. The optimizer for synthetic data construction is Adam [64] with a learning rate of 3×10^{-4}, and the number of learning iterations is set to 300, following [12].

E. Fed-TREND v.s. Baselines (RQ1)

To demonstrate the effectiveness of Fed-TREND, we compare it with several baselines in Table III. 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 Solar Energy dataset, the performance between Centralized and FedAvg is very close. This is because the data in this dataset are almost homogeneous since at any plants, the solar energy is zero at night and variations among different areas in a state are minimal. In contrast, on datasets like 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

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

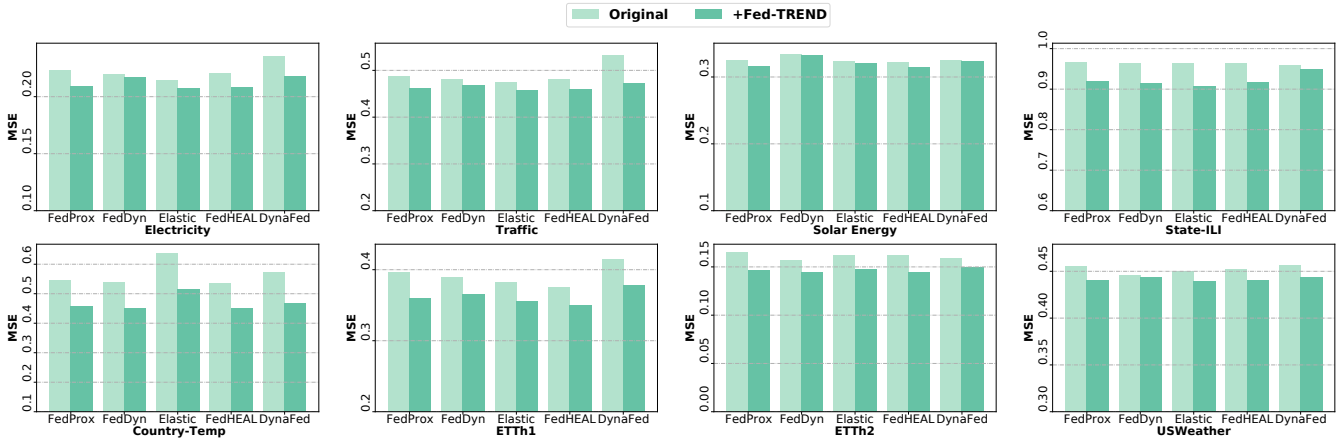

Fig. 2. The result of using Fed-TREND to improve the general federated learning frameworks.

image classification and time series forecasting. Finally, our proposed method, Fed-TREND, consistently outperforms federated learning baselines with 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.

F. 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 IV, Fed-TREND improves the performance of all tested forecasting models compared to naive federated learning. Specifically, DLinear and TSMixer achieve the best perfor-

TABLE IV
THE PERFORMANCE OF FED-TREND WITH VARIOUS STATE-OF-THE-ART TIME SERIES FORECASTING MODELS.

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

TABLE V
THE RESULTS OF ABLATION STUDY. "-CU" MEANS DOES NOT CONSTRUCT $\mathcal{D}_{ct}$ THAT ONLY CONSIDERING THE CONSISTENT UPDATED GRADIENTS, I.E., USING ALL PARAMETERS FOR $\mathcal{D}_{ct}$ CONSTRUCTION. "-$\mathcal{D}_{ct}$" AND "-$\mathcal{D}_{gt}$" MEANS REMOVE THE CORRESPONDING SYNTHETIC DATASETS.

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

mance 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.

G. 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 V. 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.

H. Hyperparameter Analysis (RQ4)

In this paper, the dataset sizes $|\mathcal{D}_{gt}|$ and $|\mathcal{D}_{ct}|$ are intuitively the two most significant hyperparameters influencing the performance of Fed-TREND. Therefore, we analyze their impact in Section V-H1 and Section V-H2, respectively. Additionally, we explore Fed-TREND's performance with different input and output lengths in Section V-H3, as these settings are

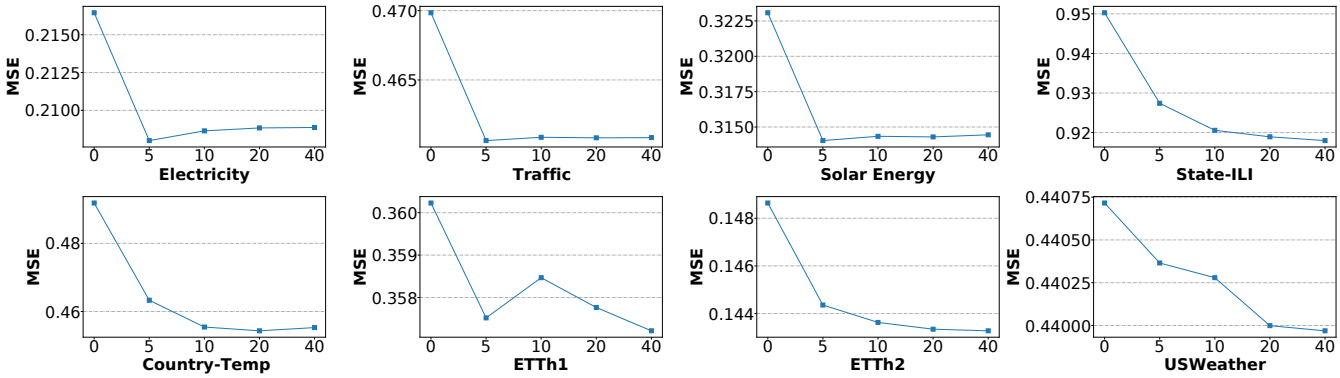

Fig. 3. The performance trend with different $|\mathcal{D}_{gt}|$.

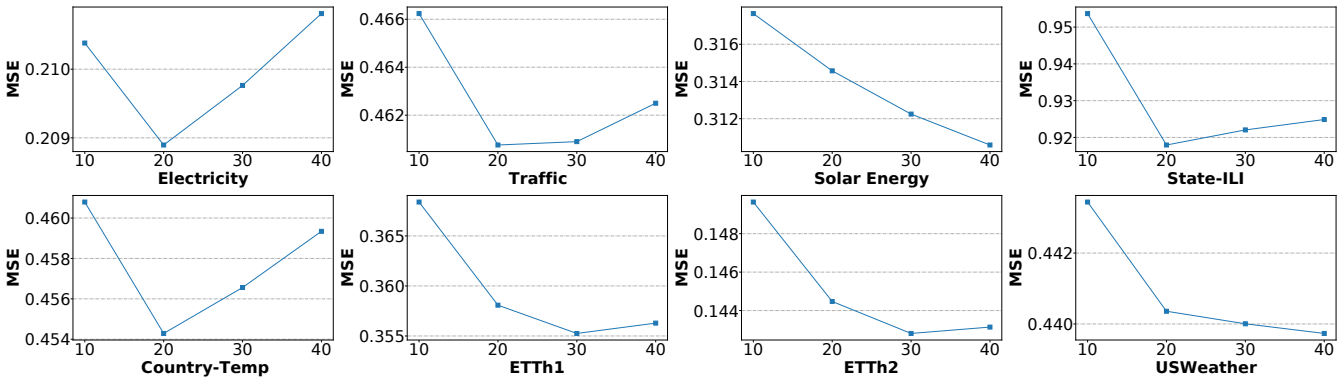

Fig. 4. The performance trend with different $|\mathcal{D}_{ct}|$.

crucial for practical time series forecasting tasks. Last but not least, we analyze the frequency of updating $\mathcal{D}_{gt}$ and $\mathcal{D}_{ct}$, i.e., the value of L_{gt} and L_{ct}'s influence in Section V-H4.

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.

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.

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 II), we only examine data lengths from 24 to 48 for this dataset.

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 actually 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 ef-

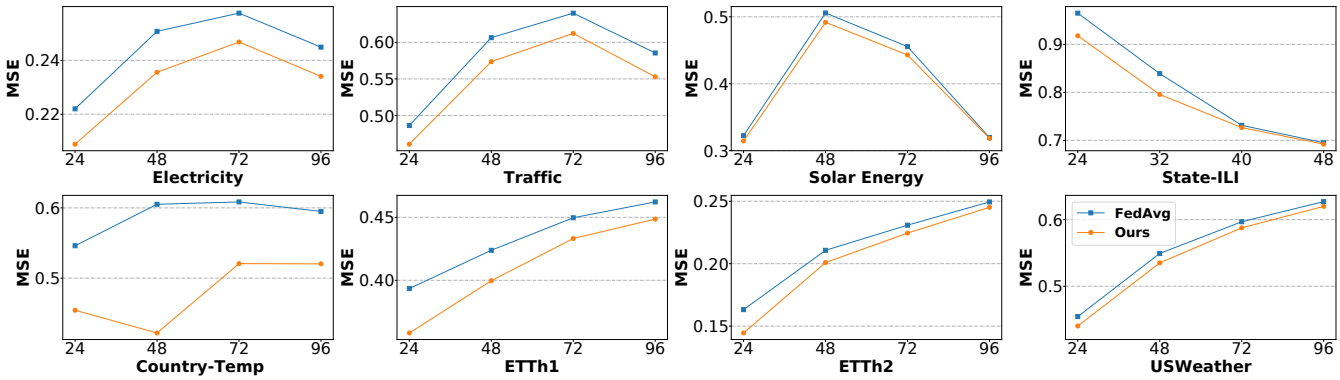

Fig. 5. The performance trend with time series data length.

fectiveness of our method across different data length settings.

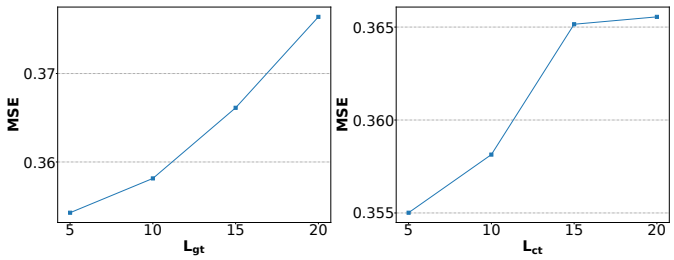

Fig. 6. The performance trend with the synthetic data construction frequency L_{gt} and L_{ct} on ETTh1. Similar trend can be observed on other datasets.

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 good balance.

I. 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 [65]. 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, 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.

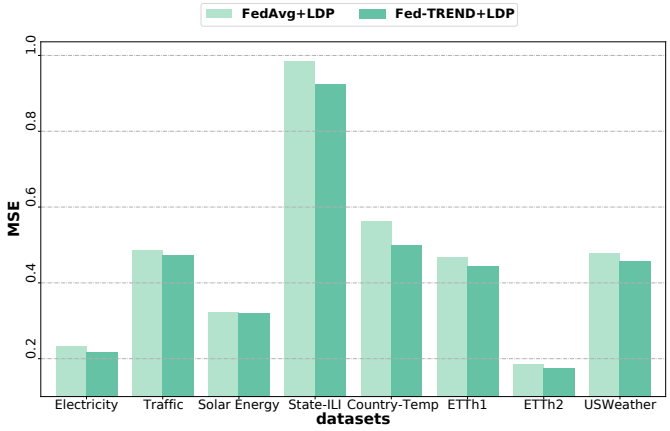

Fig. 7. The performance comparison of the base federated learning and the equipping of Fed-TREND under the context of differential privacy.

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

REFERENCES

[1] Y. Li, X. Lu, H. Xiong, J. Tang, J. Su, B. Jin, and D. Dou, "Towards long-term time-series forecasting: Feature, pattern, and distribution," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1611–1624.

[2] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[3] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," in *The Twelfth International Conference on Learning Representations*.

[4] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[5] T. Zhang, Y. Zhang, W. Cao, J. Bian, X. Yi, S. Zheng, and J. Li, "Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures," *arXiv preprint arXiv:2207.01186*, 2022.

[6] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 11 121–11 128.

[7] S.-A. Chen, C.-L. Li, S. O. Arik, N. C. Yoder, and T. Pfister, "Tsmixer: An all-mlp architecture for time series forecast-ing," *Transactions on Machine Learning Research*.

[8] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac, "Smart meter data privacy: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2820–2835, 2017.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[10] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 2022, pp. 965–978.

[11] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, "Heterogeneous federated learning: State-of-the-art and research challenges," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–44, 2023.

[12] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, "Dataset distillation by matching training trajectories," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4750–4759.

[13] R. Yu, S. Liu, and X. Wang, "Dataset distillation: A comprehensive review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[14] X. Gao, J. Yu, W. Jiang, T. Chen, W. Zhang, and H. Yin, "Graph condensation: A survey," *arXiv preprint arXiv:2401.11720*, 2024.

[15] J. Goetz and A. Tewari, "Federated learning via synthetic data," *arXiv preprint arXiv:2008.04489*, 2020.

[16] Y. Xiong, R. Wang, M. Cheng, F. Yu, and C.-J. Hsieh, "Feddm: Iterative distribution matching for communication-efficient federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 323–16 332.

[17] P. Liu, X. Yu, and J. T. Zhou, "Meta knowledge condensation for federated learning," in *The Eleventh International Conference on Learning Representations*.

[18] Y. Wang, H. Fu, R. Kanagavelu, Q. Wei, Y. Liu, and R. S. M. Goh, "An aggregation-free federated learning for tackling data heterogeneity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 26 233–26 242.

[19] Q. Wen, L. Yang, T. Zhou, and L. Sun, "Robust time series analysis and applications: An industrial perspective," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4836–4837.

[20] T. Shen, Y. Li, and J. M. Moura, "Forecasting covid-19 dynamics: Clustering, generalized spatiotemporal attention, and impacts of mobility and geographic proximity," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 2892–2904.

[21] S. Chen, G. Long, T. Shen, and J. Jiang, "Prompt federated learning for weather forecasting: toward foundation models on meteorological data," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 3532–3540.

[22] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella *et al.*, "Deep learning for time series forecasting: Tutorial and literature survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–36, 2022.

[23] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[24] E. S. Gardner Jr, "Exponential smoothing: The state of the art," *Journal of forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[25] A. C. Harvey, "Forecasting, structural time series models and the kalman filter," 1990.

[26] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[27] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

[28] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 95–104.

[29] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, pp. 1421–1441, 2019.

[30] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: a survey," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 6778–6786.

[31] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22 419–22 430, 2021.

[32] Q. V. H. Nguyen, C. T. Duong, T. T. Nguyen, M. Weidlich, K. Aberer, H. Yin, and X. Zhou, "Argument discovery via crowdsourcing," *The VLDB Journal*, vol. 26, pp. 511–535, 2017.

[33] W. Yuan, H. Yin, F. Wu, S. Zhang, T. He, and H. Wang, "Federated unlearning for on-device recommendation," in *Proceedings of the sixteenth ACM international conference on web search and data mining*, 2023, pp. 393–401.

[34] W. Yuan, L. Qu, L. Cui, Y. Tong, X. Zhou, and H. Yin, "Hetefedrec: Federated recommender systems with model heterogeneity," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 1324–1337.

[35] W. Yuan, C. Yang, L. Qu, Q. V. H. Nguyen, J. Li, and H. Yin, "Hide your model: A parameter transmission-free federated recommender system," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 611–624.

[36] W. Yuan, S. Yuan, C. Yang, N. Quoc Viet hung, and H. Yin, "Manipulating visually aware federated recommender systems and its countermeasures," *ACM Transactions on Information Systems*, vol. 42, no. 3, pp. 1–26, 2023.

[37] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[38] H. Yin, Q. Wang, K. Zheng, Z. Li, and X. Zhou, "Overcoming data sparsity in group recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 7, pp. 3447–3460, 2020.

[39] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International conference on machine learning*. PMLR, 2021, pp. 2089–2099.

[40] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in neural information processing systems*, vol. 33, pp. 2351–2363, 2020.

[41] H.-Y. Chen and W.-L. Chao, "Fedbe: Making bayesian model ensemble applicable to federated learning," in *International Conference on Learning Representations*.

[42] Q. Liu, C. Chen, J. Qin, Q. Dou, and P.-A. Heng, "Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1013–1023.

[43] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "Fedmix: Approximation of mixup under mean augmented federated learning," in *International Conference on Learning Representations*.

[44] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[45] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.

[46] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," *arXiv preprint arXiv:2111.04263*, 2021.

[47] D. Chen, J. Hu, V. J. Tan, X. Wei, and E. Wu, "Elastic aggregation for federated optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 187–12 197.

[48] Y. Chen, W. Huang, and M. Ye, "Fair federated learning under domain skew with local consistency and domain diversity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 12 077–12 086.

[49] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, "Distilled one-shot federated learning," *arXiv preprint arXiv:2009.07999*, 2020.

[50] S. Hu, J. Goetz, K. Malik, H. Zhan, Z. Liu, and Y. Liu, "Fedsynth: Gradient compression via synthetic data in federated learning," in *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*.

[51] J. Zhang, C. Chen, B. Li, L. Lyu, S. Wu, S. Ding, C. Shen, and C. Wu, "Dense: Data-free one-shot federated learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 414–21 428, 2022.

[52] B. Zheng, K. Zheng, X. Xiao, H. Su, H. Yin, X. Zhou, and G. Li, "Keyword-aware continuous knn query on road networks," in *2016 IEEE 32Nd international conference on data engineering (ICDE)*. IEEE, 2016, pp. 871–882.

[53] R. Dai, Y. Zhang, A. Li, T. Liu, X. Yang, and B. Han, "Enhancing one-shot federated learning through data and ensemble co-boosting," in *The Twelfth International Conference on Learning Representations*.

[54] R. Pi, W. Zhang, Y. Xie, J. Gao, X. Wang, S. Kim, and Q. Chen, "Dynafed: Tackling client data heterogeneity with global dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 177–12 186.

[55] N. Q. V. Hung, H. H. Viet, N. T. Tam, M. Weidlich, H. Yin, and X. Zhou, "Computing crowd consensus with partial agreement," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 1, pp. 1–14, 2017.

[56] Q. Liu, X. Liu, C. Liu, Q. Wen, and Y. Liang, "Time-ffm: Towards lm-empowered federated foundation model for time series forecasting," *arXiv preprint arXiv:2405.14252*, 2024.

[57] R. Abdel-Sater and A. B. Hamza, "A federated large language model for long-term time series forecasting," *arXiv preprint arXiv:2407.20503*, 2024.

[58] Y. Yan, G. Yang, Y. Gao, C. Zang, J. Chen, and Q. Wang, "Multi-participant vertical federated learning based time series prediction," in *Proceedings of the 8th International Conference on Computing and Artificial Intelligence*, 2022, pp. 165–171.

[59] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," *arXiv preprint arXiv:2006.05929*, 2020.

[60] G. Li, R. Togo, T. Ogawa, and M. Haseyama, "Dataset distillation using parameter pruning," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 107, no. 6, pp. 936–940, 2024.

[61] J. Du, Y. Jiang, V. Y. Tan, J. T. Zhou, and H. Li, "Minimizing the accumulated trajectory error to improve dataset distillation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 3749–3758.

[62] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[63] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*. PMLR, 2013, pp. 1139–1147.

[64] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[65] S. Zhang, W. Yuan, and H. Yin, "Comprehensive privacy analysis on federated recommender system against attribute inference attacks," *IEEE Transactions on Knowledge and Data Engineering*, 2023.