This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

A Communication-Efficient Federated Learning Scheme for IoT-Based Traffic Forecasting

Chenhan Zhang, *Student Member, IEEE*, Lei Cui, *Member, IEEE*, Shui Yu, *Senior Member, IEEE*, and James J.Q. Yu, *Senior Member, IEEE*

***Abstract*—Federated Learning (FL) is widely adopted in traffic forecasting tasks involving large-scale IoT-enabled sensor data since its decentralization nature enables data providers' privacy to be preserved. When employing *state-of-the-art* deep learning-based traffic predictors in FL systems, the existing FL frameworks confront overlarge communication overhead when transmitting these models' parameter updates since the modelling depth and breadth renders them incorporating enormous number of parameters. In this paper, we propose a practical FL scheme, namely, Clustering-based hierarchical and Two-step-updated FL (CTFed), to tackle this issue. The proposed scheme follows a *divide et impera* strategy that clusters the clients into multiple groups based on the similarity between their local models' parameters. We integrate the particle swarm optimization algorithm and devise a two-step approach for local model update. This scheme enables only one but representative local model update from each cluster to be uploaded to the central server, thus reduces the communication overhead of the model updates transmission in FL. CTFed is orthogonal to the gradient compression- or sparsification-based approaches so that they can orchestrate to optimize the communication overhead. Extensive case studies on three real-world datasets and three *state-of-the-art* models demonstrate the outstanding training efficiency, accurate prediction performance and robustness to unstable network environments of the proposed scheme.**

***Index Terms*—Federated learning, communication efficiency, graph neural networks, industrial IoTs, traffic forecasting.**

I. Introduction

In the notion of smart cities, people's travel experience significantly benefits from real-time and accurate traffic states. Traffic speed and flow are critical indicators to monitor the traffic conditions and estimate the future. Thus, a heap of research attention has been attracted to the domain of traffic forecasting. With the enhancement of massive data collection techniques in intelligent transportation systems (ITS), deep learning (DL)-based approaches, have been regarded as a preference to implement short- and long-term traffic forecasting.

Corresponding author: James J.Q. Yu (email: yujq3@sustech.edu.cn)

This work is supported by the Stable Support Plan Program of Shenzhen Natural Science Fund No. 20200925155105002, by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032, by the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation No. 2020B121201001, and by Australia ARC DP200101374 and LP190100676.

Chenhan Zhang and James J.Q. Yu are with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Chenhan Zhang is also with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. Lei Cui and Shui Yu are with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia.

Among manifold DL-based approaches, the recently popular Graph Neural Networks (GNN)-based approaches are recognized as the *state-of-the-art* due their powerful capacity of spatial dependency learning that are well-suited to the feature exploiting of transportation networks. GNN-based traffic predictors are extensively adopted in *de facto* industrial applications [1], [2]. Take the collaboration of Google Maps and Deepmind as an example [2], GNN-based estimated time of arrival (ETA) predictors are developed based on information from Google Maps platform and applied to multiple cities in the world, where the accuracy of real-time ETA has been improved up to 50%.

Most GNN-based models are trained in a centralized manner, where the traffic data are collected by multiple providers such as governmental transportation organizations (e.g., California Department of Transportation) and authorized companies (e.g., Google and NavInfo). Nevertheless, the traffic data may incorporate personal privacy (e.g., GPS trajectories and plate numbers) or business-sensitive information (e.g., allocation of the roadside unit (RSU)), which raises privacy concerns. In the learning community, Federated Learning (FL) has emerged as a technology for solving the aforementioned centralized training privacy issues of ML and DL. Specifically, FL achieves privacy protection for data providers by enabling distributed ML/DL training without original data transfer and keeping the providers' data locally. In ITS domain, extensive applications have shown the efficacy of FL in different tasks, including vehicular networking [3], traffic control [4], and traffic forecasting [5], [6].

While most optimization methods for FL systems are developed and tested considering conventional ML/DL models, it is not surprising to generalize FL to GNN-based models. However, the massive studies on GNNs and FL indicate a practical concern. The majority of FL systems adopt the averaging algorithm (i.e., FedAvg [7]) to develop the global model, which requires multiple participants (e.g., data providers and workers[1]) to upload their local models to the central server. This will lead to a considerable network communication overhead and storage costs for the central server. Unlike the conventional DNN models that handle low-dimensional grid-like data, GNNs that handle high-dimensional and sparse graph-structured data are more computationally intractable and, meanwhile, involve more parameters with the same model depth. The adoption of the latter will significantly enlarge the

[1]In crowdsourced FL systems, the local participants are usually termed as "workers" which may only train the local model over data received from the central server (task initiator) without providing personal data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

overall computing and communication costs to FL systems [8]. Furthermore, for existing data collection systems in ITS, the data collecting nodes, such as RSUs, and mobile devices (e.g., smartphones), may run in areas under unstable networks (e.g., Wi-Fi). Such network environments pose a challenge for achieving real-time data processing in any FL systems. Existing research towards the communication optimization of FL includes gradient compression [9]–[11], asynchronous update [12], [13], etc. However, as mentioned above, most of these approaches are developed and tested considering simple ML/DL tasks and models, whose generalization capacity and effect to practical tasks equipped with state-of-the-art models are sceptical. Furthermore, there are few practical communication-efficient solutions that merge GNN-based traffic forecasting approaches into FL. These approaches either engender a multitude of information loss in the compression process or are not applicable to large-scale and real-time systems.

To bridge the research gap, in this paper, we propose a practical FL scheme, namely, Clustering-based hierarchical and Two-step-updated FL (CTFed). In CTFed, we follow the idea of dispersed FL and propose a hierarchical FL architecture, and accordingly devise a clustering approach to cluster the clients into multiple groups. Particularly, the proposed clustering rule is based on the clients' local model parameter similarity, and we attempt to put the clients with similar model parameters into the same cluster. Then we integrate the particle swarm optimization (PSO) algorithm and devise a two-step approach for local model update. Particularly, this scheme attempts to find the best local model by fitness evaluation, which only requires the client to upload a measured value regarding the fitness to the server deployed in each cluster instead of the model parameters. Consequently, the scheme of CTFed enables only one representative local model update from each cluster to be uploaded to the central server, which significantly reduces the communication overhead of the model updates transmission in the FL.

The highlights of this paper are summarized below:

- We propose a communication-efficient scheme, named CTFed, for reducing the communication overhead of FL system equipped with state-of-the-art deep learning-based traffic predictors. The proposed CTFed is orthogonal to the prevailing gradient compression- or sparsification-based approaches.
- In the proposed CTFed, we propose a model parameters similarity-based clients clustering approach that can cluster the client with similar model parameter. Furthermore, we devise a two-step local model update approach is devised based on the PSO algorithm.
- The proposed CTFed is divide-and-conquer, which enables only one representative local model update from each cluster to be uploaded to the central server in the model uploading phase of FL. The incorporated model parameters similarity-based clients clustering approach and two-step local model update approach can orchestrate to guarantee the contribution ability of the uploaded single model.
- A series of comprehensive case studies on three state-of-the-art GNN-based traffic predictors and three real-world traffic datasets are conducted to demonstrate the efficacy of the proposed CTFed scheme.

The rest of this paper is organized as follows. In Section II, we review the related literature on traffic speed forecasting approaches and communication optimization approaches for FL. Section III gives a basic formulation of the traffic speed forecasting problem and the federated learning scenario to be investigated in this paper. We elaborate on the proposed CTFed scheme in Section IV. Section V presents the results and discussion of the case studies. Section VI discusses the traits of the proposed scheme, and this paper is concluded in Section VII with a summary of potential future studies.

II. Related Work

A. State-of-the-art Approaches for Traffic Forecasting

The great success of deep learning has been witnessed in the traffic forecasting domain. Deep learning-based models such as Recurrent Neural Networks (e.g., Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)), and Convolutional Neural Networks (CNN) are employed in the literature [1], [14]. Due to the discovery of the spatial correlation among traffic nodes' latent influence on time-series traffic data, a great many research efforts have been made to develop spatial feature capturing algorithms in traffic forecasting approaches [15]. In [16], the authors proposed an approach that treats the transportation information as an image and adopts CNN to exploit the latent spatial correlation from the image. Yu *et al.* [17] proposed a hybrid spatial-temporal model by integrating CNN and LSTM, in which the output of CNN are adopted as the input for LSTM hierarchically.

Albeit the effect in spatial feature learning, CNN-based approaches [16], [17] can only process grid-like spatial structures which are Euclidean while the traffic data is sampled from traffic networks that are non-Euclidean. To overcome this problem, researchers start to use graphs for traffic modeling by treating the irregular traffic networks as graphs incorporating topology of different traffic nodes (e.g., road intersections) [18], [19]. Moreover, with the Graph Deep learning (GDL) techniques springing up, researchers extend the conventional neural networks to graph domain, i.e., Graph Neural Networks (GNN) – Graph Convolutional Network (GCN) [20] for CNN, Graph Attention Networks (GAT) [21] for attention mechanism to name a few. ITS researchers employ GNNs to learn the spatial correlations of traffic graphs. Li *et al.* [22] utilized directional diffusion convolution on the road network to capture the spatial correlations and GRU for time-series dependency. Yu *et al.* [23] adopted spectral-domain GCN and Gated CNN to extract the spatial and temporal correlations, respectively. Do *et al.* [24] computed spatial and temporal attentional factors in traffic graph data to extract the spatial-temporal dependencies. Some other state-of-the-art GNN-based traffic forecasting approaches can be seen in [25], [26].

Moreover, traffic forecasting is an ideal investigated application for the proposed scheme in this paper. The prevailing graph representation-based deep learning in this area usually

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

involve complex modeling that introduce forecasting models incorporating large number of model parameters.

B. Communication Optimization of FL

To optimize the communications of FL systems, some efforts have been made, and the consequential approaches can be categorized into two classes, namely, gradient compression and asynchronous update [27]. In [11], the authors employed gradient quantization and encoding techniques in the stochastic gradient descent (SGD) optimizer to shrink the uploaded gradients from local models. Wu *et al.* [28] proposed an error compensated quantization-based SGD algorithm to improve the training efficiency. Specifically, the algorithm utilizes accumulated quantization error to accelerate the training convergence. Lin *et al.* [29] used gradient sparsification that makes local participants only send important gradients to the central server for efficient communication. Stich *et al.* [30] and Liu *et al.* [31] studied the top-k sparsification algorithm from theoretical level and application level, respectively. Shi *et al.* [32] and Wangni *et al.* [10] presented rigorous mathematical proof of the convergence developed by gradient sparsification-based optimizers. Kang *et al.* [33] indicated that in addition to communication optimization, gradient sparsification also relieves the inference attack issues by only sharing partial gradient information. Meanwhile, some studies also pointed out the limitation of gradient compressed- or sparsified-based approaches' practicality limitation. In [34], the authors demonstrated the instances that gradient compression-based methods cannot converge. In [35], the authors found only a small number of successful cases that can provide salient speedup over optimized synchronous data-parallel training among their large number of tests on gradient compression methods.

Asynchronous update introduces another communication optimization solution where local participants can communicate with the central server by asynchronous updates in a FL system, which can achieve a reduction in the computing waiting time of the local participants [12], [13], [36].

This work proposes a series of schemes for easing overlarge communication burden in FL systems applied to traffic forecasting tasks using deep models. The proposed FL scheme is communication-efficient. Also, it guarantees accurate forecasting performance, circumventing the adoption of any gradient quantization or sparsification approaches that may degrade the performance of collaboratively-trained models when the model's architecture is complex and the number of involved parameters for feature learning is large. Nonetheless, the proposed approach is also orthogonal to the gradient quantization or sparsification approaches; this opens the possibility of their integration for a better solution in future studies.

III. Preliminary

In this work, the involved state-of-the-art traffic predictors are graph representation-based. Therefore, in this section, we first formulate the problem of traffic forecasting problem on graphs. Then, the investigated FL scenario of traffic forecasting with GNN-based models is presented. For the sake of clarity, we summarize the frequently used symbols of this paper in Table I.

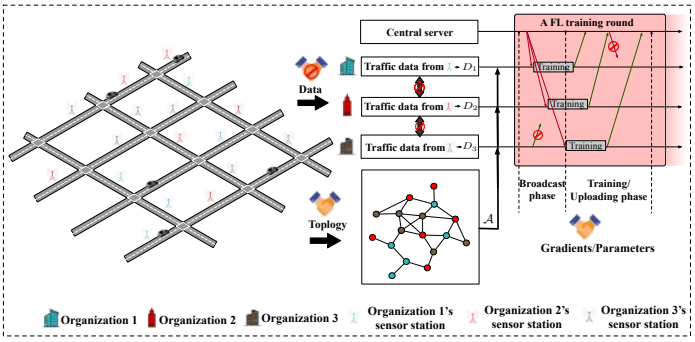

Fig. 1. The investigated FL scenario of this work[2].

A. Traffic Forecasting Problems

Traffic forecasting on graphs denotes the inference process of traffic data using observed ones sampled from several topology-correlated nodes (e.g., sensor stations) deployed in the traffic networks. The definition is given as below.

Definition 1 Traffic forecasting on graphs. *We represent a transportation network as an undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes and each of them is defined as a road segment, $\mathcal{E}$ is the set of edges, and $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix of $\mathcal{G}$. For all $v_i, v_j \in \mathcal{V}$, we have $[\mathcal{A}_{i,j}] = 1$ if v_i and v_j are connected and otherwise $[\mathcal{A}_{i,j}] = 0$, where $[\mathcal{A}_{i,j}]$ is the entry of $\mathcal{A}$ that represents the connectivity between node v_i and node v_j. This is a common formulation among spatial-temporal traffic forecasting, see examples in [22], [23], [37]. The traffic data observed on $\mathcal{G}$ is denoted as a graph-wide feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times\mathcal{F}}$ where $\mathcal{F}$ is the dimension of involved features (i.e., traffic records) of each node. Let vector $\mathbf{X}^t \in \mathbb{R}^{|\mathcal{V}|}$ denote the traffic data observed at time $t \in \mathcal{F}$, the objective is to learn a predictor $f(\cdot)$ that can develop graph-wide traffic predictions $\hat{\mathbf{X}}^{t+1}, \hat{\mathbf{X}}^{t+2}, \ldots, \hat{\mathbf{X}}^{t+s}$ in the following s time stamps (i.e., the prediction time horizon is s), given historical traffic observations of T stamps (i.e., the historical time horizon is T) $\mathbf{X}^{t-T+1}, \mathbf{X}^{t-T+2}, \ldots, \mathbf{X}^{t}$.*

B. Federated Learning Scenario

In this paper, we consider a realistic FL scenario for the proposed scheme. Particularly, we deploy advanced GNN-based models that can exploit the spatial correlation of the traffic graph in the FL system to achieve more accurate traffic forecasting. To enable any local organization to fully leverage spatial correlations, all clients have unrestricted access to the entire traffic network's topological knowledge but must keep their particular traffic data private. This is to say that the topology privacy regarding each organization is not considered in this paper. Additionally, this study is based on the assumption that there are no overlapping sensor stations between any two organizations, and thus the data. This refers to a common privacy assumption among the literature, see [38]–[40] for some instances. The preceding definitions of the central server

[2]Note that this figure only presents the working scenario and related assumptions for the proposed approach, and the readers who attempt to know the systematic structure and working procedures of the proposed approach can refer to Fig. 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

TABLE I
SUMMARY OF SYMBOLS

Symbol	Explanation
$\mathcal{G}$	Traffic graph.
$\mathcal{V}$	Set of road segments (nodes) of $\mathcal{G}$.
$\mathcal{E}$	Set of road intersections (edges) of $\mathcal{G}$.
$\mathcal{A}$	Adjacency matrix of $\mathcal{G}$.
$\mathbf{X}$	Graph-wide traffic record matrix.
$\mathcal{F}$	Dimension of traffic record.
C_i	Organization i.
M	Number of organizations.
D_i	Database of C_i.
f_i	Local model of C_i.
θ_i	Model parameters of f_i
$\mathrm{sim}_{i,j}$	Cosine similarity between θ_i and θ_j.
k	Number of clusters.
$\vec{R}$	Lower-dimensional vector of θ.
$\vec{v}_i$	Speed of particle i in PSO algorithm.
$\vec{x}_i$	Position of particle i in PSO algorithm.
$\vec{\mathrm{pb}}_i$	Personal best of particle i in PSO algorithm.
$\vec{\mathrm{gb}}$	Global best of all particles in PSO algorithm.
ω	Inertia constants for particles in PSO algorithm.
ϕ	Acceleration constants for particles in PSO algorithm.

and local clients and in the investigated FL scenario can be seen as below.

Definition 2 Central server. *The central server is a third-party and trustworthy data center in the cloud. It selects a proper state-of-the-art GNN-based model, f, for the task. In the broadcasting phase of FL system, the central server distributes a group of copies of the adopted model, $\{f_1, f_2, \ldots, f_M\}$, and the adjacency matrix of the graph-represented entire traffic network $\mathcal{G}$, $\mathcal{A}$, to the local clients. In the uploading phase of FL system, the central server receives locally trained models from the clients and updates the global model.*

Definition 3 Local clients. *The entire transportation network is divided and possessed by multiple organizations*[3] *(e.g., companies, governments, and individuals). Let $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ denote the organization set where M is the number of organizations. Each of the organizations operates several sensor stations, i.e., several nodes of the graph, and their respective node sets are denoted by $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_M\}$. Let $\mathcal{D} = \{D_1, D_2, \ldots, D_M\}$ denote the database set of the organizations, each of which stores traffic data collected from the organization's respectively operated sensor stations, which satisfies $D_i \cap D_j = \emptyset \ \forall\, C_i, C_j \in \mathcal{C}$. The organizations treat the distributed models $\{f_1, f_2, \ldots, f_M\}$ as local model and train them simultaneously and locally utilizing the stored training data from $\mathcal{D}$.*

Furthermore, this paper only focuses on the *synchronous update* that the organizations communicate with the server regularly for model updates as the adoptions demonstrated in [38], [41]. This paper assumes that all the participants of the FL system are *honest*. An *honest* participant will strictly execute the FL's operating rules and will not perform any actual malicious behavior. An illustration of the introduced scenario can be seen in Fig. 1.

[3]In this paper, "organization" and "client" are used interchangeably.

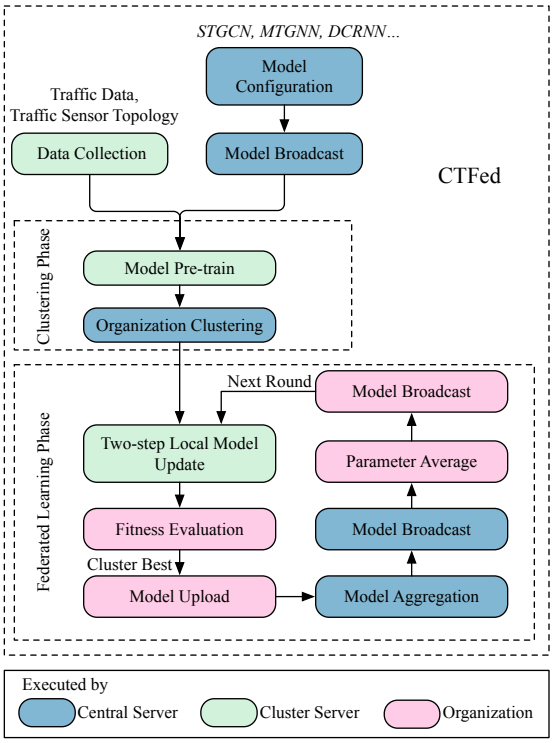

Fig. 2. The schematic of CTFed.

IV. METHODOLOGY

This section first briefly introduces the architecture of the proposed FL scheme. Subsequently, we will elaborate on the proposed technical components involved in the scheme and discuss how it achieves communication-efficient and accurate FL for GNN-based traffic forecasting.

A. Scheme Overview

As illustrated in Fig. 2, the proposed scheme involves two phases, namely, **clustering phase** and **FL phase**. In the organization clustering phase, following a "divide-and-conquer" strategy, the organizations are clustered into multiple clusters by the similarity between their pre-trained models' parameters. This clustering algorithm will be presented in Section IV-B. In the FL phase, we propose a novel Particle Swarm Optimization (PSO)-based algorithm for local model training, which will be detailed in Section IV-C along with the system's communication protocol.

B. Parameters Similarity-based Organizations Clustering

Due to the homogeneity of the datasets' time series and spatial correlations, for some organizations, the differences among the learnt parameters of their models are small. Aggregating homogeneous models does not contribute to the development of a generic global model, and it also imposes an unnecessary communication overhead by transmitting such homogeneous model parameters to the central server, especially when a large number of participating organizations is involved [42]. To cope with this issue, inspired by the similarity-based clustering algorithms proposed in [41] and the hierarchical and dispersed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

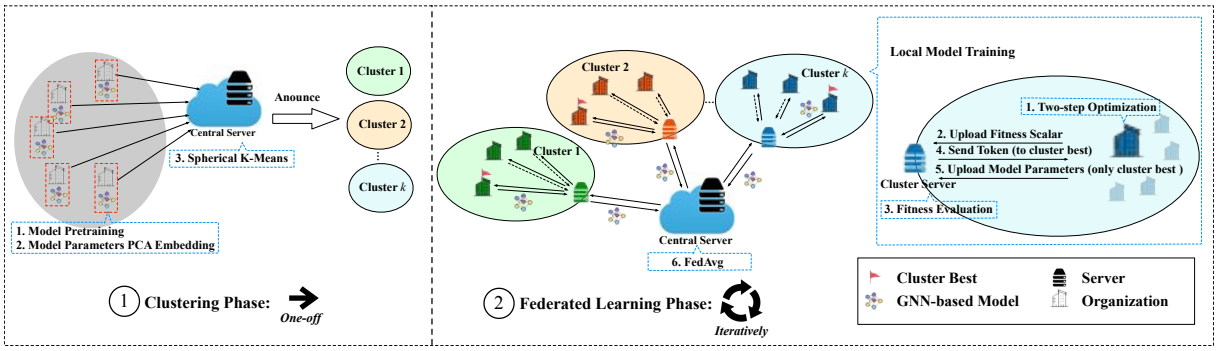

Fig. 3. An illustration of CTFed. In the Federated Learning Phase, and the solid line represents the model parameters transmission while the dash line represents the fitness scalar transmission.

FL structure introduced in [43], we propose a parameter similarity-based approach to clustering the organizations.

In this approach, the clustering decision is determined by the similarity of organizations' model parameters. Specified to the high dimensional characteristics of the GNN-based model parameters, we adopt Principal Component Analysis (PCA) [44] to implement a dimensionality reduction on the model parameters and the spherical K-means algorithm, i.e., the K-means algorithm with *cosine similarity* to achieve the clustering. Note that the dimensionality reduction is executed within the client while the clustering is implemented by the central server with the received reduced model parameter vectors. Specifically, to obtain the model parameters for similarity computing, we first define a *pre-learned* model parameters for each organization. Particularly, we pre-train the local models of each organization. For organization C_i and its data $\mathbf{X}_i \in \mathbb{R}^{|\mathcal{V}|\times\mathcal{F}}$, we use a portion of its historical data $\mathbf{X}_i^* \in \mathbb{R}^{|\mathcal{V}|\times\mathcal{F}^*}$ which is randomly sampled from the time series windows, satisfying $\mathcal{F}^* \ll \mathcal{F}$ is the sampled dataset[4]. From our offline Monte Carlo simulations, the influence of the randomness in this step on the final clustering decision is not significant. Subsequently, $\mathbf{X}_i^*$ is used to train the local model, and the trained model parameters θ_i^* is regarded as the pre-learned model parameters. Treating the model parameters as a high-dimensional vector $\theta_i^* \rightarrow \vec{\theta}_i^h$, we use PCA to project the high-dimensional $\vec{\theta}_i^h$ into a vector $\vec{R}_i$ which is in a lower-dimensional space. Particularly, we do not set a pre-defined targeted dimension; instead, we empirically set the summative variance of all individual principal components $\sum\sigma^2 = 0.9$ to avoid information loss.

Then, following the spherical K-means algorithm, given a number of clusters k satisfying $k \ll M$, we randomly select k organizations as the initial cluster centroids. For each organization, we seriatim compute the similarity between its model parameters vector and the cluster centroids' ones. Particularly, for any two organizations C_i and C_j, the cosine similarity between their lower-dimensional model parameter vectors $\vec{R}_i$ and $\vec{R}_j$ is defined as

$$\mathrm{sim}_{i,j} := \mathrm{sim}(\vec{R}_i, \vec{R}_j) := \frac{\sum_{u=1}^{l} r_{i,u} r_{j,u}}{\sqrt{\sum_{u=1}^{l} r_{i,u}^2}\sqrt{\sum_{u=1}^{l} r_{j,u}^2}}, \quad (1)$$

[4]The number of random sampling time points, i.e., F^*, are set the same for each organization.

where l is the dimension of the model parameter vector; $r_{i,u}$ and $r_{j,u}$ denote the scalars of $\vec{R}_i$ and $\vec{R}_j$ respectively, and u is the index. By the nature of the cosine similarity metric, the scoring between any two total factor vectors is symmetric; thus, we have $\mathrm{sim}(\vec{R}_i, \vec{R}_j) = \mathrm{sim}(\vec{R}_j, \vec{R}_i)$. Whereafter, we assign the organization to the cluster whose centroid has the highest parameter similarity with it. This optimization proceeds iteratively until the optimal solution is found, i.e., the assignments no longer change. The detailed pipeline of this clustering algorithm and involved communication procedure is illustrated in Algorithm 1 and Fig. 3.

C. Particle Swarm Optimization-based Two-step Parameters Updating

In the traditional FL paradigm, FedAvg [7] as a Secure Parameters Aggregation Mechanism (SPAM) is usually adopted to perform synchronous optimization, which privately aggregates the outputs of local models from clients to update a global model on the central server. As thus, the process of FedAvg consists of two parts, i.e., client-end update and server-end update. The client-end update describes that the client trains the server-distributed model over its local dataset, which presents a commonplace model training. The server-end update of FedAvg describes that the central server takes an average of the resulting models parameters θ_i, which can be formulated as

$$\theta \leftarrow \sum_{C_i \in \mathcal{C}^*} p_i \theta_i, \ \mathcal{C}^* \subseteq \mathcal{C}, \quad (2)$$

where $\mathcal{C}^*$ denotes the set of clients who participate in the update, $\mathcal{C}^* = \mathcal{C}$ denotes there is a *full client participation* and otherwise *partial client participation*; $p_i \geq 0$ is the weight of the i-th client and $\sum_{C_i \in \mathcal{C}^*} p_i = 1$. We consider $\forall p_i = \frac{1}{|\mathcal{C}^*|}$ as unweighted averaging and otherwise weighted averaging.

Limitations of FedAvg: FedAvg requires the clients to upload the complete model parameters to the central server. For *state-of-the-art* GNN-based spatial-temporal prediction models, the number of incorporated parameters can be enormous as shown in Table II. In practical FL systems, the communication bandwidth and network delay become the bottlenecks of transmitting a large number of gradients between the cloud server and clients since the transmission will result in expensive communication overhead [42]. Additionally, as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

Algorithm 1: Parameters Similarity-based Organizations Clustering

Input: Organization set $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$, traffic data records $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_M\}$, adjacency matrix of traffic graph $\mathcal{A}$, number of clusters k, GNN-based model f, gradient optimizer $\nabla\mathcal{L}(\cdot,\cdot,\cdot)$, learning rate η, number of training epochs E, mini-batch size B.

Output: Cluster set $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$.

CentralServer: `// running on the central server`

Initialize the GNN-based model f
Broadcast copies of f and $\mathcal{A}$ to $\{C_1, C_2, \ldots, C_M\}$
if $\{\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_M\}$ *are received* **then**
  Initialize cluster set $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$
  Randomly select k organizations $\mathcal{O} = \{O_1, O_2, \ldots, O_k\}$ as the initial cluster centroids
  while *not convergence* **do**
    foreach $C_j \in \mathcal{C}$ **do**
      foreach $O_i \in \mathcal{O}$ **do**
        $\mathrm{sim}_{i,j} \leftarrow \mathrm{sim}(\vec{R}_{i\leftarrow\mathrm{centroid}}, \vec{R}_j)$ via (1)
      $O_u \leftarrow \arg\max([\mathrm{sim}_{i,j}])$
      Assign C_i to P_u
    foreach $P_i \in \mathcal{P}$ **do**
      $\vec{R}_{i\leftarrow\mathrm{centroid}} \leftarrow \frac{1}{|P_i|}\sum_{C_j\in P_i}\vec{R}_j$
    Update $\mathcal{P}$
  Announce $\mathcal{P}$ to **Organizations**;

Organization: `// running on the organization` C_i

$\mathbf{X}_i^* \leftarrow \mathrm{RandomSample}(\mathbf{X}_i)$
$\mathcal{B} \leftarrow$ (split $\mathbf{X}_i^*$ into batches of size B)
if f_i *is received* **then**
  foreach *epoch* $e \in E$ **do**
    foreach *batch* $b \in \mathcal{B}$ **do**
      $\theta_i^* \leftarrow \theta_i^* - \eta \cdot \nabla\mathcal{L}(\theta_i^*, \mathcal{A}, b)$
  $\vec{R}_i = \mathrm{PCA}_{\sum\sigma^2}(\vec{\theta}_i^h)$
  Upload $\vec{R}_i$ to **CentralServer**

a synchronous optimization approach, FedAvg may suffer from serious "straggler's effect" (i.e., everyone waits for the slowest one) in real-world applications. For example, if there are thousands of clients in the FL system, a portion of clients go slow due to various practical reasons (e.g., unstable network environment, inferior computing power, and passive participation). The requirement of full client participation asks the central server must wait for these "stragglers", which lengthens the training cycle. While partial client participation can mitigate this problem to some extent, the aggregation result may be biased due to fewer clients' participation.

To overcome the problems mentioned above that happened to FedAvg, in the proposed FL scheme, we devise a two-step parameter update approach incorporating particle swarm optimization (PSO) algorithms integrated into FedAvg's client-end update (i.e., local model training).

[5]The numbers of model parameters are counted on the modelling of PeMSD7 dataset.

[6]The presented ARIMA model is the vanilla one that only considers single time-series prediction.

TABLE II
INCORPORATED PARAMETERS OF TRAFFIC PREDICTION MODELS

Model	GNN-based	No. parameters[5]
ARIMA[6]	–	684
CNN-LSTM [17]	–	5 596
T-GCN [45]	✓	13 257
STGAT [19]	✓	100 779
Graph WaveNet [46]	✓	246 089
STGCN [23]	✓	330 345
MTGNN [25]	✓	433 721
DCRNN [22]	✓	495 518

Originally proposed in [47], PSO is an iterative optimization algorithm that combines extensive global optimization capabilities with ease of implementation, rapid convergence, and robustness. With simple mathematical operators, PSO is of less computational overhead with regards to both speed and memory [48]. The PSO algorithm incorporates a group of particles, namely, a swarm. Each particle represents a solution to the global problem, which has its specified position and speed in the solution space. Particularly, the speed determines the next position that the particle will move to. The fitness value is to evaluate the performance of the particle. To find the global optimal solution, the particles interact with each other repeatedly to tweak themselves to the optimal. The PSO algorithm in an iteration can be mathematically formulated as

$$\begin{cases} \vec{v}_i^{\tau+1} = \omega\cdot\vec{v}_i^{\tau} + \vec{U}(0,\phi_1)\cdot(\vec{\mathrm{pb}}_i - \vec{x}_i^{\tau}) \\ \qquad\qquad + \vec{U}(0,\phi_2)\cdot(\vec{\mathrm{gb}} - \vec{x}_i^{\tau}) \\ \vec{x}_i^{\tau+1} = \vec{x}_i^{\tau} + \vec{v}_i^{\tau+1} \end{cases}, \tag{3}$$

where τ denotes the τ-th iteration of the PSO algorithm; ω denotes the inertia constant; $\vec{v}_i^{\tau}$ and $\vec{x}_i^{\tau}$ are the speed and position of the particle i at iteration τ, respectively; $\vec{\mathrm{pb}}_i$ denotes the "personal best", which is the best solution that the particle i can individually develop heretofore; $\vec{\mathrm{gb}}$ denotes the "global best", which is the best solution that all particles can reach heretofore; ϕ_1 and ϕ_2 are the acceleration constants for $\vec{\mathrm{pb}}_i$ and $\vec{\mathrm{gb}}$, respectively; $\vec{U}(0,\phi_1)$ and $\vec{U}(0,\phi_2)$ denotes two vectors of random values uniformly sampled from $[0,\phi_1]$ and $[0,\phi_2]$ respectively, which are initialized at each iteration for each particle.

The idea of PSO is leveraged in the local model training of the proposed scheme, in which we devise a two-step parameter update algorithm.

Step 1: in a local training epoch, we first use PSO to update the model parameters, and the process can be formulated as

$$\nabla\mathrm{PSO}: \begin{cases} \vec{v}_{i,j}^{\tau+1} = \omega\cdot\vec{v}_{i,j}^{\tau} + \vec{U}(0,\phi_1)\cdot(\vec{\mathrm{pb}}_{i,j} - \vec{v}_{i,j}^{\tau}) \\ \qquad\qquad + \vec{U}(0,\phi_2)\cdot(\vec{\mathrm{cb}}_j - \vec{v}_{i,j}^{\tau}) \\ \vec{\theta}_{i,j}^{\tau+1} = \vec{\theta}_{i,j}^{\tau} + \vec{v}_{i,j}^{\tau+1} \\ \vec{\mathrm{pb}}_{i,j} = \vec{\theta}_{i,j}^{\tau+1} \end{cases}, \tag{4}$$

where i, j denotes the i-th client in the j-th cluster; $\vec{\mathrm{cb}}_j$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

Algorithm 2: FL phase of CTFed

Input: Cluster set $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, inertia constant ω, acceleration constants ϕ_1 and ϕ_2, number of global training rounds E_{global}, learning rate η, number of local training epochs E_{local}, number of batches B.

Output: Global model parameters $\vec{\theta}^{\text{global}}$.

CentralServer: `// running on the central server`

Initialize the GNN-based model f
Broadcast copies of f to $\{P_1, P_2, \ldots, P_k\}$
foreach *epoch* $e \in E_{\text{global}}$ **do**
  if $\{\vec{\theta}_1^{\text{best}}, \vec{\theta}_2^{\text{best}}, \ldots, \vec{\theta}_k^{\text{best}}\}$ *are received* **then**
    $\vec{\theta}^{\text{global}} \leftarrow$ execute FedAvg via (7)
    Broadcast $\vec{\theta}^{\text{global}}$ to $\{P_1, P_2, \ldots, P_k\}$

ClusterServer: `// running on cluster server` P_j

if f_j *(or* $\vec{\theta}^{\text{global}}$*) is received* **then**
  if $\vec{\text{cb}}^*_j$ *exists* **then**
    Broadcast $\vec{\text{cb}}_j$ derived by (8) to $C_{i,j} \in P_j$
  else
    Broadcast f_j (or $\vec{\theta}^{\text{global}}$) as $\vec{\text{cb}}_j$ to $C_{i,j} \in P_j$
if *all* $s_{i,j}$ *are received* **then**
  $C_{b,j} \leftarrow \arg\min [s_{i,j}]$ via (6)
  Send token to $C_{b,j}$
if $\vec{\theta}^{\text{ts}}_{b,j}$ *is received* **then**
  Upload $\vec{\theta}^{\text{ts}}_{b,j}$ as $\vec{\theta}^{\text{best}}_j$ to central server and store a copy of $\vec{\theta}^{\text{best}}_j$ as $\vec{\text{cb}}^*_j$

Organization: `// running on organization` $C_{i,j}$

$\mathcal{B} \leftarrow$ (split $\mathbf{X}_{i,j}$ into batches of size B)
if $\vec{\text{cb}}_j$ *is received* **then**
  Initialize $\vec{v}_{i,j}$, $\vec{\theta}_{i,j}$, ω, ϕ_1, ϕ_2, $\vec{\text{pb}}_{i,j}$
  $\vec{\theta}^{\text{pso}}_{i,j} \leftarrow$ update $\vec{\theta}_{i,j}$ using PSO via (4)
foreach *epoch* $e \in E_{\text{local}}$ **do**
  foreach *batch* $b \in \mathcal{B}$ **do**
    $\vec{\theta}^{\text{pso}}_{i,j} \leftarrow \vec{\theta}^{\text{pso}}_{i,j} - \eta \cdot \nabla \mathcal{L}(\vec{\theta}^{\text{pso}}_{i,j}, \mathcal{A}, b)$
$\vec{\theta}^{\text{ts}}_{i,j} \leftarrow \vec{\theta}^{\text{pso}}_{i,j}$
$\mathbf{X}^*_{i,j} \leftarrow \text{RandomSample}(\mathbf{X}_{i,j})$
$s_i \leftarrow \vec{\theta}^{\text{ts}}_{i,j}, \mathbf{X}^*_{i,j}$ via (6)
Upload s_i to **ClusterServer**
if *token is received* **then**
  Upload $\vec{\theta}^{\text{ts}}_{i,j}$ to **ClusterServer**

represents the optimal model parameter of cluster P_j before this iteration. We denote the pso-updated model parameter by $\vec{\theta}^{\text{pso}}_{i,j}$.

Step 2: Subsequently, we further update $\vec{\theta}^{\text{pso}}_{i,j}$ by a conventional gradient descent approach in a mini-batch manner, which can be formulated as

$$\nabla\text{GD}: \vec{\theta}^{\text{pso}}_{i,j} \leftarrow \vec{\theta}^{\text{pso}}_{i,j} - \eta \cdot \nabla \mathcal{L}(\vec{\theta}^{\text{pso}}_{i,j}, \mathcal{A}, \mathbf{X}_{i,j}). \tag{5}$$

To find the optimal local model of a cluster, we introduce a fitness scalar s among the clients, which is obtained by the loss of the two-step updated $f_{i,j}$ on a section of randomly sampled data denoted by $\mathbf{X}^*_{i,j}$ whose size is predefined by q, and the loss is measured by Mean Absolute Percentage Error (MAPE) without percentage operation. After receiving all the fitness scalars from clients, the cluster server then performs a *fitness evaluation* that quests for the smallest one among the scalars denoted by $s_{b,j}$, and correspondingly, $C_{b,j}$ denotes the organization which has the smallest scalar. The whole process of fitness evaluation can be formulated as

$$b := \arg\min_i s = \frac{1}{|\mathbf{X}^*_{i,j}|} \sum_{u=1}^{|\mathbf{X}^*_{i,j}|} \left| \frac{X^*_u - \hat{X}^*_{u \leftarrow \vec{\theta}^{\text{ts}}_{i,j}}}{X^*_u} \right|, \tag{6}$$

where X^*_u denotes a data sample of $\mathbf{X}^*_{i,j}$ and u is the index; $\hat{X}^*_{u \leftarrow \vec{\theta}^{\text{ts}}_{i,j}}$ represents the prediction developed by the model with two-step-updated parameter $\vec{\theta}^{\text{ts}}_{i,j}$. Consequently, the cluster server identify organization $C_{b,j}$ as the "cluster best" and asks $C_{b,j}$ to upload its model parameters (by sending a token) and 1) store a copy of it temporarily as $\vec{\text{cb}}^*_j$ and 2) upload another copy of it to the central server. Additionally, we introduce a *second-string* mechanism to remedy the transmission failure. Specifically, if the communication between the "cluster best" client and cluster server fails, the cluster server will request the client that has the second-highest fitness scalar to upload its model parameters, and so forth.

According the divide-and-conquer strategy of the proposed scheme, a sub-global model with $\vec{\theta}^{\text{best}}_j$ is first developed for different clusters by fitness evaluation. In this step, we directly select the local model developing the smallest fitness scalar as the sub-global model rather than aggregating all local models, since PSO can calibrate the local model with global optimization locally to develop "global-optimized" local models. From the perspective of communication optimization, only one client is asked to upload the model parameters in such a case, which obviously reduces the overall communication cost. Then, we aggregate the sub-global models from different clusters in the central server to update the global model using FedAvg, which can be formulated as

$$\vec{\theta}^{\text{global}} = \frac{1}{k} \sum_{j \in k} \vec{\theta}^{\text{best}}_j, \tag{7}$$

where $\vec{\theta}^{\text{global}}$ denotes the model parameters of the global model. It is worth noting that we do not let the clients who are cluster best to send their model parameters directly to the central server is because this requires additional communication channels (clients – central server) which may raise additional security concerns in practice [49].

In the model broadcasting step, instead of directly assign the global update to the clients, we first let the cluster server to average the previously stored $\vec{\text{cb}}^*_j$ and the received global model's parameters $\vec{\theta}^{\text{global}}$ by

$$\vec{\text{cb}}_j = \frac{1}{2}(\vec{\text{cb}}^*_j + \vec{\theta}^{\text{global}}). \tag{8}$$

Then, the cluster server broadcasts $\vec{\text{cb}}_j$ to included clients for the next iteration of local model optimization. In this way, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

TABLE III
SUMMARY OF PEMSD4, PEMSD7, AND METR-LA DATASETS.

Dataset	No. sensors	No. time stamps	Sampling period
PeMSD4	307	16 992	01/01/18-02/28/18
PeMSD7	228	12 672	05/01/12-06/30/12
METR-LA	207	12 302	03/01/12-06/30/12

TABLE IV
TRAINING CONFIGURATION FOR GNN-BASED MODELS.

Model	Optimizer	Learning rate	Weight decay	Dropout
STGCN	RMSprop	5×10^{-5}	0	0.2
T-GCN	Adam	2×10^{-6}	5×10^{-4}	0.1
MTGNN	Adam	10^{-3}	10^{-4}	0.1

send an aggregation of the global update and the cluster best to the clients, which can make the PSO algorithm develop a relatively smooth optimization following the previous global epoch [50].

For a better understanding of the audience, we present the procedures and communication protocol of the FL phase of CTFed detailedly in Algorithm 2.

V. EXPERIMENTS

In this paper, we propose CTFed as a FL scheme for collaborative traffic forecasting tasks. To fully evaluate the efficacy of the proposed scheme, we conduct a series of comprehensive case studies on three real-world traffic datasets. Particularly, we first investigate the accuracy and training efficiency of forecasting speed using the proposed scheme. Subsequently, we conduct a hyperparameter sensitivity test to illustrate how to design the scheme to yield the best system performance. Then, an ablation study is conducted to evaluate the critical components of CTFed. Finally, we exhibit the performance of CTFed under different FL settings.

A. Dataset and Configurations

In our case studies, three real-world datasets are adopted for investigation, i.e., PeMSD4, PeMSD7, and METR-LA. PeMSD4 and PeMSD7 are two public datasets published by California Department of Transportation[7], which contains traffic speed data of Bay area and Greater Los Angeles Area. METR-LA incorporates traffic information collected from loop detectors in the highway of Los Angeles County by LA-Metro [22]. The traffic graph topology of the three datasets is developed based on sensor stations' distance, referring to [23]. The adoption of the three public datasets is to provide a fair comparison for other researchers in the traffic forecasting community. We summarize the three datasets in Table III. The data points in all three datasets are with a 5-min sampling interval. Z-score is adopted to normalize the speed values, and linear interpolation is used to recover missing data points. For each dataset, the training, validation, and test sets are correspondingly constructed for supervised learning, each containing 60%, 20%, and 20% of all data, respectively. In all case studies of traffic speed forecasting, the past time window is 60 minutes (i.e., 12 timestamps), and we use them to predict the speed in the next 45 minutes (i.e., nine timestamps). MAPE and RMSE are used as the accuracy metrics, which are defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{X_i - \hat{X}_i}{X_i}\right| \times 100\%, \tag{9a}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2}, \tag{9b}$$

where X_i and $\hat{X}_i$ are observed and predicted traffic speeds at time i, respectively. In particular, MAPE is considered as a preferable one (see [51], [52] for examples).

We include three *state-of-the-art* GNN-based models in our case studies:

- **STGCN** [23]: A GNN-based and CNN-based model which integrates graph convolutions with 1D convolutions.
- **T-GCN** [45]: A GNN-based and RNN-based model which incorporates graph convolutions with GRU.
- **MTGNN** [25]: A GNN-based and CNN-based model which leverages mix-hop propagation, adaptive graph, and dilated inception to exploit spatial-temporal dependencies.

The architectures of the three models follow the best settings accessible in their corresponding literature, and their respective training configurations are set as shown in Table IV.

Unless otherwise noted, for a fair comparison, we set the global epoch $E_{\text{global}} = 30$ and the mini-batch size $B = 50$ for all case studies. Additionally, we set the number of local training epochs $E_{\text{local}} = 1$ for all the FL-based approaches to make them compared with the centralized training approach fairly. Additionally, to simulate the FL training scenario for the proposed scheme, we construct the respective dataset of different organizations. In particular, we first partition the whole traffic graph into M sub-graphs for M organizations using Metis partitioner [53]. Then we select the traffic data of included sensors (nodes) in a sub-graph and construct the dataset for the corresponding organization. Subject to the computing resource, we let $M = 8$ for all FL simulations. For CTFed, we by default let $k = 3$ and $\omega, \phi_1, \phi_2 = 0.1, 1, 4$.

CTFed and the baseline approaches are implemented with PyTorch using half-precision (i.e., $\mathrm{float16}$) tensors. All case studies are conducted on a computing server with two Intel Xeon E5 CPUs, and eight nVidia GTX 2080 Ti GPUs are used for computing acceleration.

B. Learning Performance Comparison

In this case study, we configure the aforementioned three GNN-based models to the proposed scheme and evaluate the traffic forecasting accuracy performance on the three datasets, respectively. Moreover, we compare the performance between the proposed scheme and five baseline approaches:

- **Centralized**: Centralized training algorithm (i.e., training without FL).

[7]https://pems.dot.ca.gov/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

TABLE V
PREDICTION ACCURACY COMPARISON.

MAPE	PeMSD4			PeMSD7			METR-LA		
	STGCN	T-GCN	MTGNN	STGCN	T-GCN	MTGNN	STGCN	T-GCN	MTGNN
Centralized	4.75%	5.53%	4.72%	6.93%	5.53%	7.02%	7.31%	8.47%	7.36%
FedAvg	4.79%	5.53%	4.73%	6.90%	8.04%	6.99%	7.48%	8.32%	7.45%
FedP	4.84%	5.72%	4.77%	7.09%	9.15%	7.09%	7.53%	8.73%	7.60%
FedQSGD	8.31%	13.46%	5.73%	15.56%	27.25%	11.62%	12.32%	19.73%	10.45%
FedTopK	16.70%	13.55%	11.51%	29.02%	27.12%	23.27%	22.07%	20.15%	17.38%
CTFed	4.80%	5.62%	4.78%	6.96%	8.07%	6.98%	7.69%	8.40%	7.55%

RMSE	PeMSD4			PeMSD7			METR-LA		
	STGCN	T-GCN	MTGNN	STGCN	T-GCN	MTGNN	STGCN	T-GCN	MTGNN
Centralized	4.78	5.35	4.93	5.11	5.35	5.33	5.33	5.81	5.44
FedAvg	4.82	5.36	4.90	5.13	5.79	5.29	5.31	5.76	5.40
FedP	4.89	5.47	5.00	5.27	6.23	5.38	5.42	5.89	5.48
FedQSGD	7.16	10.06	5.55	8.30	13.27	7.19	6.90	10.05	6.38
FedTopK	11.44	10.31	8.92	13.76	13.26	11.45	11.00	10.32	9.11
CTFed	4.87	5.39	4.90	5.16	5.79	5.21	5.38	5.78	5.35

- **FedAvg**: Unweighted FedAvg algorithm with *full client participation* as introduced in Eq. (2).
- **FedP**: Unweighted FedAvg algorithm with *partial client participation* in each global epoch as introduced in Eq. (2). For a fair comparison with CTFed, we let $|\mathcal{C}^*| = 3$ in our simulations.
- **FedQSGD**: FedAvg algorithm integrating with gradient compression [11]. QGSD 4-bit is adopted in our simulations.
- **FedTopK**: FedAvg algorithm integrating with k-sparsification for gradient sparsification [30]. $k = 1$ for sparsification is adopted in our simulations[8].

For FedQSGD and FedTopK, we implement them following the algorithms in the respective literature with minuscule non-algorithmic changes.

Table V summarizes the prediction accuracy results of the proposed scheme and baselines. Figure 4 presents the corresponding learning curves, and Figure 5 illustrates the data transmission volume for the FL-based approaches. From the results, a few conclusions can be derived. First, the GNN-based models can obtain satisfying prediction accuracy with the proposed CTFed scheme, which is close to FedAvg and Centralized. This is credited to the efficacy of the two approaches in CTFed. First, the clustering approach makes the model parameters of the clients in the same cluster more possibly similar; thus, any one of them can be regarded as a representative one for others in the cluster. On this basis, the proposed two-step local model updating approach integrated PSO algorithm can efficiently calibrate the local parameter with global parameter, and the conventional GD algorithm can further lead to a correct convergence direction, which orchestrates a promising learning performance. Furthermore, as shown in Figure 5, CTFed has lower communication overhead than FedAvg while achieving comparable accuracy performance. Comparing CTFed and FedP, while their communication overhead is close, the latter's convergence rate is not as fast as CTFed. The fewer clients' participation in each global epoch makes FedP harder convergence; while in CTFed, the equally fewer participated local models are updated by the proposed approaches in CTFed that the models to be aggregated by the center serval are more representative than those of FedP.

While FedQSGD and FedTopK approaches can incur demonstrably fewer communication overhead (See Figure 5) due to their adopted gradient compression and sparsification algorithms, in our simulations, we can find obviously poor convergence and accuracy performance developed by these two approaches. Especially for the FedTopK approach, the model does not converge, and the MAPE metric quadruples than the approaches without using gradient sparsification when adopting STGCN on PeMSD7. This is due to that remarkably spatial-temporal information learned and embedded by the model parameters are undermined during the processes of gradient compression/sparsification when using these approaches. Although we find promising results presented in their original literature [11], [30], it is worth noticing that these results are developed using simpler or Euclidean modeling-based approaches (e.g., logistic regression and CNN-based models). That is to say unless there is further well-pointed optimization, these gradient compression- or sparsification-based approaches cannot achieve promising results using GNN-based models on at least complex tasks such as traffic forecasting.

Last but not least, on all the three GNN-based models and three real-world datasets, the proposed scheme can obtain outstanding results, demonstrating the proposed scheme's generalization ability.

C. Ablation Test

*1) **Hyperparameter Sensitivity Test**:* The selection of hyperparameters included in the scheme is another significant factor for FL performance. Particularly, we first assess the im-

[8]The FedAvg is with full client participation for FedQSGD and FedTopK.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

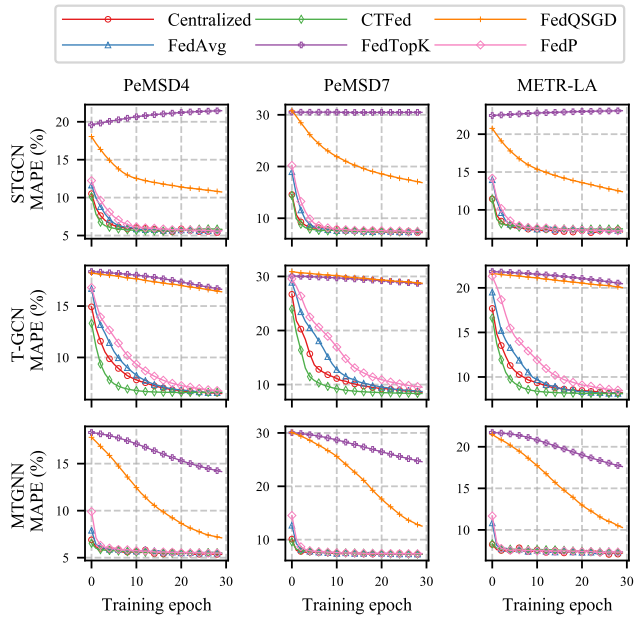

Fig. 4. Learning curves of the proposed scheme and baselines.

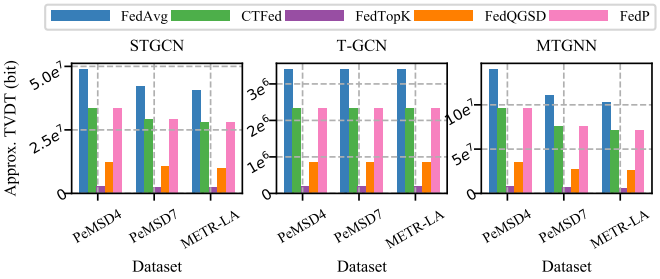

Fig. 5. Total volume of data transmission (TVDT) in a global epoch.

pact of the number of clusters k on the learning performance[9]. In this experiment, we set $k = 2, 3, 4$ for CTFed and compare the learning performance under this group of settings.

From the simulation results shown in Table VI and Figure 6, we have the following observations. First, the smaller number of involved clusters, the more difficult the learning curves converge. However, the final accuracy performance under different k is with little difference. This can be explained by the fact that a larger number of clusters can accelerate

[9]Note that since the performance results on three datasets demonstrate a similar pattern, for conciseness without loss of generality, we only present the FL performance on PeMSD7 dataset in the following simulations.

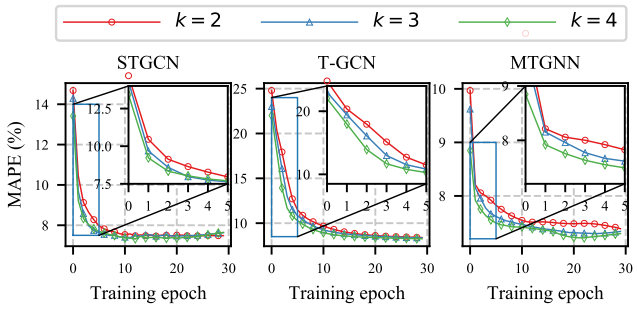

Fig. 6. The sensitivity of learning performance to the number of clusters k.

TABLE VI
THE SENSITIVITY OF ACCURACY PERFORMANCE TO THE NUMBER OF CLUSTERS k.

	STGCN		T-GCN		MTGNN	
	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
$k = 2$	7.10%	5.25	8.13%	5.82	7.07%	5.20
$k = 3$	7.10%	5.25	8.09%	5.81	7.01%	5.21
$k = 4$	7.10%	5.25	8.09%	5.81	7.10%	5.21

TABLE VII
ACCURACY PERFORMANCE COMPARISON BETWEEN CTFED AND VARIANTS.

	STGCN		T-GCN		MTGNN	
	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
OG	7.10%	5.22	8.13%	5.81	7.07%	5.19
AC	7.09%	5.22	8.09%	5.81	6.93%	5.19
WC	7.11%	5.23	8.40%	5.98	7.09%	5.31
FP	28.41%	14.04	28.75%	14.05	28.48%	13.98

the convergence speed since the global model can aggregate learned information from more local models according to the algorithm of the proposed scheme. Nevertheless, as the number of iterations increases, the global model can aggregate information from more local models even with a small k, and consequently, it will be more generic to develop accurate results. Furthermore, recalling the advantage of the proposed scheme as presented in Section IV, we know that the smaller k, the less communication overhead the FL system develops. These results support the previous statement that the CTFed is capable of achieving the trade-off between accuracy and communication overhead.

In addition to the number of clusters k, it is also interesting to investigate the impact of the hyperparameters of the PSO algorithm in the proposed scheme, i.e., inertia constant ω, acceleration constants ϕ_1 and ϕ_2. Specifically, we let $\omega \in \{0.05, 0.1, 0.2\}$, $\phi_1 \in \{0.5, 1, 2, 3, 4\}$ and $\phi_2 \in \{2, 3, 4, 5, 6\}$ to conduct a grid-search to identify the best portfolio. The results are visualized in Figure 7.

The simulation results imply that CTFed is limitedly sensitive to the different selection of hyperparameters in the PSO algorithm. In particular, the MAPE ranges for the three models are $6.90\% - 7.20\%$ (STGCN), $8.06\% - 8.19\%$ (T-GCN), and $6.84\% - 7.14\%$ (MTGNN). While the grid-search results do not demonstrate a clear pattern to determine a better hyperparameters portfolio, such fine-tuning work can slightly improve the FL performance.

*2) **Derived Approaches Test**:* In the proposed scheme, two core approaches, namely, parameters similarity-based organizations clustering and PSO-based two-step local model parameters update, are orchestrated to achieve communication-efficient and accurate collaborative traffic forecasting. In this case study, a comprehensive ablation test is conducted to validate the contribution of these components to the overall FL performance. Specifically, several variants are constructed based on the original CTFed design as follows:

- CTFed-AC (Adaptive-Clustering): The proposed clustering approach is implemented iteratively in the FL phase

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3132363, IEEE Internet of Things Journal

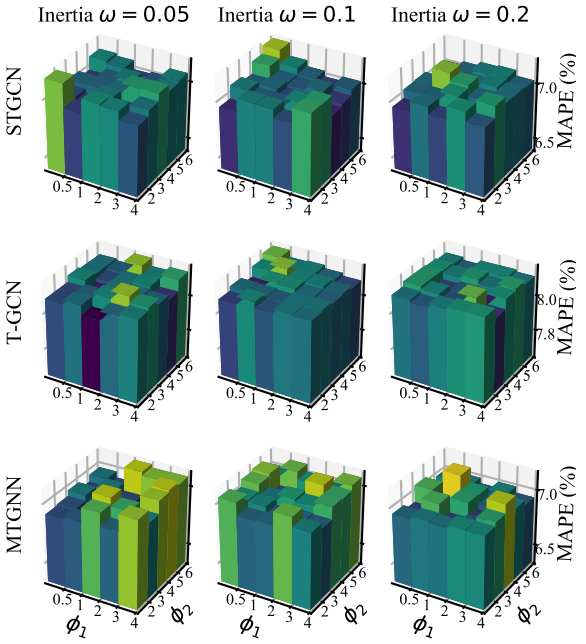

Fig. 7. The sensitivity of PSO parameters ω, ϕ_1, ϕ_2 to accuracy performance.

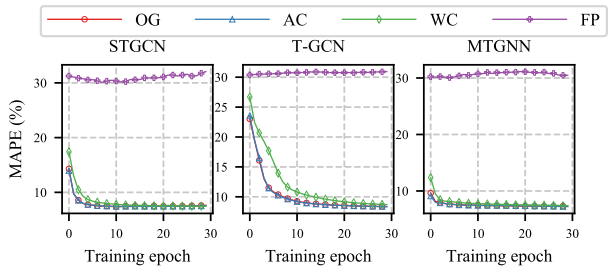

Fig. 8. Learning performance comparison among CTFed and its variants.

by the central server.

- CTFed-WC (Without-Clustering): The clustering and FedAvg algorithms are removed. The clients send their fitness scalars to the central server directly, and the central server implements the fitness evaluation and requests the best local model's parameter for global model updating.
- CTFed-FP (Full-PSO): In the proposed two-step update for local models, the GD optimization is removed, and only the PSO is used (i.e., one-step optimization).

We use CTFed-OG to denote the original CTFed scheme in this case study. The simulation results are shown in Table VII and Figure 8.

Comparing CTFed-OG and CTFed-AC, we find that the performance difference is minuscule; however, the latter's design renders it costing more communication and computing overhead due to its repetitive clustering computation. This result clearly indicates that the design of one-off clustering in the proposed approach can provide the same level of performance as the iterative one with less overhead. Comparing CTFed-OG and CTFed-WC, it can be observed that the latter's convergence speed is slower than the former. This is because the clustering and FedAvg algorithms in the proposed scheme can speed up the global model's generalization, which does not have a counterpart in the CTFed-WC. Additionally, we can observe that the variant CTFed-FP does not converge. Without the assist of gradient descent, PSO cannot exclusively steer the model in the desired direction of convergence, which explains the necessity of a two-step update strategy in the proposed scheme. To conclude, these results support the adoption of constituting components of CTFed.

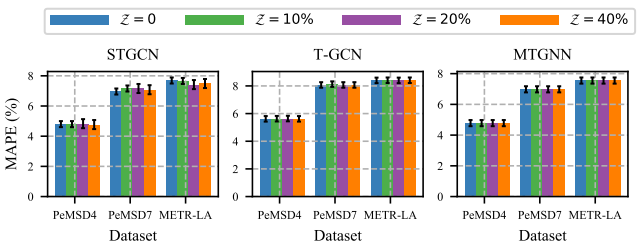

Fig. 9. Accuracy performance with different transmission failure rates.

D. Robustness to Volatile Network Transmission

In real-world FL-based traffic forecasting tasks, unstable network environments is an inevitable problem that may deteriorate the quality of data transmissions between different participants [54], [55], and consequently, render the system developing inferior accuracy performance. In this case study, we carry out a preliminary test on the accuracy performance of CTFed under an unstable network transmission environment. Specifically, we simulate the unstable network transmission environment by randomly dropping the data transmitted from the client to the cluster server. Denoting the failure rate (i.e., dropping rate) as $\mathcal{Z}$, we evaluate the FL performance with $\mathcal{Z} = 0\%, 10\%, 20\%, 40\%$ where $\mathcal{Z} = 0\%$ represents the CTFed's performance in ideal condition.

The results are summarized in Figure 9. We can observe the accuracy performance remains on the same level when the failure rate increase from 0% to 40%. This test demonstrates the robustness of CTFed to an unstable network transmission environment, which is contributed by the failure-tolerance capability of the fitness evaluation strategy. In particular, recalling the communication protocol of the proposed scheme as introduced in Section IV-C, only the model parameters from one client of each cluster is required to be uploaded to the central server. If some clients' data transmissions are failed, for those who are the "cluster best" client, the introduced second-string will request the second-best client to upload the model parameters, which mitigates the possible issue.

VI. Discussion

In this section, we present a discussion from three perspective, 1) communication, 2) generalization and applicability, and 3) security and privacy, to highlight the merits as well as the limitations of the proposed scheme.

A. Communication

As discussed in Section IV-C, communication bandwidth is an inevitable bottleneck for FL since scores of clients attempt to communicate with the server. Furthermore, the communication volume of gradient/parameter is large, especially for

the investigated GNN-based models for spatial-temporal traffic forecasting.

CTFed mitigates these issues in two aspects. First, the fitness evaluation step on the clustering server does not require the clients to send the whole updated model parameters but a scalar (i.e., the fitness scalar which is introduced in Section IV-C), which reduces the data transmission volume significantly. Referring to the number of training parameters of GNN-based models that used to be transmitted in FL systems as listed in Table II, the reduction in data transfer volume in a single transmission can reach 100,000 times for the proposed scheme. Second, the hierarchical structure of CTFed disperses the communication pressure to the cluster servers. For example, assuming that there are M clients participating in the FL and k clusters obtained by CTFed, the number of concurrent communications to be handled by the central server is reduced from M to k where $k \ll M$.

B. Generalization and Applicability

In this work, CTFed is proposed to reduce the communication overhead in the FL systems that configure state-of-the-art deep learning-based traffic predictors. While the communication overhead problem is investigated around the models for traffic forecasting, this problem is not peculiar to traffic forecasting, which also exists in other scenarios. Thus, the proposed scheme can be used for a broader range of FL-based applications. For those scenarios which have, but not limited to, the following characteristics, CTFed are well-suited:

- The number of the parameters in the collaborative model is huge, and thus the advantages of CTFed can be fully realized.
- Horizontal federated learning. It uses datasets with the same feature space across all clients, which means that any two clients have the same set of features; thus, the homogeneity between different clients' data and locally trained models exists.

C. Security and Privacy

While the security and privacy problems are out of this paper's scope, the proposed FL scheme can lower the risk of the clients' models or data being compromised. Considering a rough scenario of gradient/parameter leakage attack in the model updating phase, assume that there are M clients participating in the FL, the security protection level is the same for different transmission channels, and the probability that the parameters are compromised in each transmission is α. For those conventional FL frameworks, the overall probability that the FL framework is compromised is $P_{\mathrm{FL}} = \sum^{M} \alpha = M\alpha$. Comparatively, in CTFed with k clusters, according to Algorithm 2, the overall probability that the FL framework is compromised is $P_{\mathrm{CTFed}} = 2\sum^{k} \alpha = 2k\alpha$. As $k \ll M$, we have $P_{\mathrm{CTFed}} \ll P_{\mathrm{FL}}$. Therefore, without regard to adopting any defense mechanisms or cryptographic techniques, the proposed FL scheme can reduce the risk that the clients' models or data are compromised.

Furthermore, the *divide-and-conquer* strategy adopted in the proposed FL scheme can help avoid cascading failure. Considering the scenario that the global model is poisoned, for the conventional FL frameworks, all the clients will be influenced inevitably. However, for CTFed, the broadcasted global model update can be further examined on the cluster servers (e.g., there is a proactive system configured in the cluster server to assess if the global model is poisoned), if one of the cluster servers finds that the model is poisoned, it can terminate further broadcasting the global model update to the clients belonging to its cluster, which saves these clients. We will include the above discussion in the future work. We also expect and welcome the interested readers to conduct in-depth investigation together.

VII. Conclusion

In this paper, we propose a practical scheme to reduce the communication overhead of FL system equipped with state-of-the-art deep learning-based traffic predictors. Compared with the existing FL frameworks, the proposed CTFed employs a dispersed and hierarchical architecture by using a clustering approach to cluster the clients into different groups based on their parameters similarity. We also devise a two-step approach for the local models' update in the proposed scheme, which follows the idea of particle swarm optimization. This scheme enables the model parameters from only one client of each cluster to be uploaded to the central server, which remarkably cuts down on the communication cost of model update transmissions in the FL.

To evaluate the performance of the proposed scheme, we conduct a series of comprehensive case studies with three *state-of-the-art* GNN-based predictors on three real-world traffic datasets. Compared with baselines, the proposed scheme can achieve the trade-off between efficient model communication and accurate model performance in the investigated FL scenario. We also conduct an ablation test to assess the efficacy of the components of CTFed and the hyperparameters sensitivity to the prediction performance.

In the future, on the one hand, we will extend our study to more complex traffic conditions (e.g., prediction considering traffic accident factors and communication delay caused by extreme weather). On the other hand, we will conduct a theoretical analysis of the proposed scheme and investigate its applicability on other tasks beyond traffic forecasting.

References

[1] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, "Enhancing transportation systems via deep learning: a survey," *Transportation research part C: emerging technologies*, vol. 99, pp. 144–163, 2018.

[2] O. Lange and L. Perez, "Traffic prediction with advanced graph neural networks," 2020.

[3] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[4] G. Hua, L. Zhu, J. Wu, C. Shen, L. Zhou, and Q. Lin, "Blockchain-based federated learning for intelligent control in heavy haul railway," *IEEE Access*, vol. 8, pp. 176830–176839, 2020.

[5] Y. Liu, S. Zhang, C. Zhang, and J. James, "Fedgru: Privacy-preserving traffic flow prediction via federated learning," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[6] C. Zhang, S. Zhang, J. James, and S. Yu, "Fastgnn: A topological information protected federated learning approach for traffic speed forecasting," *IEEE Transactions on Industrial Informatics*, 2021.

[7] 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*, pp. 1273–1282, 2017.

[8] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou, and Y. Dai, "Towards efficient large-scale graph neural network computing," *arXiv preprint arXiv:1810.08403*, 2018.

[9] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, "Ternary compression for communication-efficient federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[10] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 1306–1316, 2018.

[11] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.

[12] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 21–28, Springer, 2018.

[13] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2019.

[14] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 499–508, 2016.

[15] A. M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148–163, 2018.

[16] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, Apr. 2017.

[17] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, Jun. 2017.

[18] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 485–492, 2019.

[19] C. Zhang, J. J. Yu, and Y. Liu, "Spatial-temporal graph attention networks: A deep learning approach for traffic forecasting," *IEEE Access*, vol. 7, pp. 166246–166256, 2019.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[22] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[23] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3634–3640, 2018.

[24] L. N. Do, H. L. Vu, B. Q. Vo, Z. Liu, and D. Phung, "An effective spatial-temporal attention based neural network for traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 108, pp. 12–28, 2019.

[25] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020.

[26] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1234–1241, 2020.

[27] Y. Liu, R. Zhao, J. Kang, A. Yassine, D. Niyato, and J. Peng, "Towards communication-efficient and attack-resistant federated edge learning for industrial internet of things," *arXiv preprint arXiv:2012.04436*, 2020.

[28] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error compensated quantized sgd and its applications to large-scale distributed optimization," in *International Conference on Machine Learning*, pp. 5325–5333, PMLR, 2018.

[29] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018.

[30] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *arXiv preprint arXiv:1809.07599*, 2018.

[31] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020.

[32] S. Shi, K. Zhao, Q. Wang, Z. Tang, and X. Chu, "A convergence analysis of distributed sgd with communication-efficient gradient sparsification.," in *IJCAI*, pp. 3411–3417, 2019.

[33] J. Kang, Z. Xiong, C. Jiang, Y. Liu, S. Guo, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Scalable and communication-efficient decentralized federated edge learning with multi-blockchain framework," in *International Conference on Blockchain and Trustworthy Systems*, pp. 152–165, Springer, 2020.

[34] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *International Conference on Machine Learning*, pp. 3252–3261, PMLR, 2019.

[35] S. Agarwal, H. Wang, S. Venkataraman, and D. Papailiopoulos, "On the utility of gradient compression in distributed training systems," *arXiv preprint arXiv:2103.00543*, 2021.

[36] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, 2020.

[37] K. Guo, Y. Hu, Z. Qian, H. Liu, K. Zhang, Y. Sun, J. Gao, and B. Yin, "Optimized graph convolution recurrent neural network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 1138–1149, 2020.

[38] Y. Liu, J. J. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *arXiv preprint arXiv:2003.08725*, 2020.

[39] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "Pmf: A privacy-preserving human mobility prediction framework via federated learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–21, 2020.

[40] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, 2020.

[41] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[42] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[43] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8866–8870, IEEE, 2020.

[44] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[45] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[46] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 1907–1913, AAAI Press, 2019.

[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[48] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, "Parameters optimization of deep learning models using particle swarm optimization," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1285–1290, IEEE, 2017.

[49] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms

and performance analysis," *IEEE Transactions on Information Forensics and Security*, 2020.

[50] S. Bansal, E. Ghaderi, C. Puglisi, and S. Gupta, "Neural-network based self-initializing algorithm for multi-parameter optimization of high-speed adcs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 106–110, 2020.

[51] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3940–3951, 2019.

[52] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[53] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," tech. rep., Computer Science & Engineering (CS&E) Technical Reports, 1997.

[54] T. Wang, X. Cao, and S. Wang, "Self-adaptive clustering and load-bandwidth management for uplink enhancement in heterogeneous vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5607–5617, 2019.

[55] C. Zhao, L. Cai, and P. Cheng, "Stability analysis of vehicle platooning with limited communication range and random packet losses," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 262–277, 2020.