FedLoc: Federated Learning Framework for Data-Driven Cooperative Localization and Location Data Processing

Feng Yin, Zhidi Lin, Qinglei Kong, Yue Xu, Deshi Li, Sergios Theodoridis, Shuguang (Robert) Cui

In this overview paper, data-driven learning model-based cooperative localization and location data processing are considered, in line with the emerging machine learning and big data methods. We first review (1) state-of-the-art algorithms in the context of federated learning, (2) two widely used learning models, namely the deep neural network model and the Gaussian process model, and (3) various distributed model hyper-parameter optimization schemes. Then, we demonstrate various practical use cases that are summarized from a mixture of standard, newly published, and unpublished works, which cover a broad range of location services, including collaborative static localization/fingerprinting, indoor target tracking, outdoor navigation using low-sampling GPS, and spatio-temporal wireless traffic data modeling and prediction. Experimental results show that near centralized data fitting- and prediction performance can be achieved by a set of collaborative mobile users running distributed algorithms. All the surveyed use cases fall under our newly proposed *Federated Localization (FedLoc)* framework, which targets on collaboratively building accurate location services without sacrificing user privacy, in particular, sensitive information related to their geographical trajectories. Future research directions are also discussed at the end of this paper.

***Index Terms*—Cooperation, Data-driven models, Distributed processing, Federated learning, Gaussian processes, Location services, User privacy.**

I. Introduction

WITH the explosion of data and the ever-increasing computing power, we have witnessed nowadays the popularity of machine learning models and algorithms which are data-driven. In principal, with more data, an underlying complex system/dynamic/regression function can be closely approximated. However, when the data size increases beyond a limit, both the scale of the model and the computational complexity of an associated learning algorithm can become computationally tough. For instance, the computational complexity for training a Gaussian process model scales cubically with the data size [1]. This renders the required computational load for sophisticated data-driven learning models prohibited for practical cases.

The recently proposed federated learning framework [2] has received a lot of attention, as it enables a large-scale machine learning models to be trained jointly by a large number of mobile users through cooperation. Actually, there exist various similar works before the federated learning, for instance [3], [4], but federated learning emphasizes more on the following aspects: (1) non-i.i.d. data; (2) unbalanced local data size; (3) large number of local users; (4) limited communication; and (5) data privacy [2]. It deserves to highlight that federated learning is a promising technical solution to solve the ever-increasing concerns about the loss of user privacy and to meet the ever-stringent data protection regulations world-wide, for instance, the General Data Protection Regulation (GDPR) implemented by the European Union in 2018. Federated learning has triggered various potential applications in the sectors of smart medicine, finance, and next-generation wireless communications [5], [6], [7]. In this paper, we extend federated learning to a new application sector, namely target localization and location-related services.

Target localization is meant to provide an estimate of the desired position as accurate as possible. There exist a plethora of state-of-the-art techniques for static target localization, target tracking, navigation, and interested readers can refer to [8], [9], [10] and the references therein for more information. Most of these techniques rely on empirical, parametric transition and measurement models, which can be regarded as an individual abstract of human experience, thus they may severely mismatch the underlying mechanism in complicated environments such as office, shopping mall, museum, etc. However, directly learning from a huge volume of historical data may help alleviate such a model mismatch and improve the positioning accuracy even further.

Apart from the traditional localization service, a new type of location related services have emerged in the recent years under the umbrella of smart cities, namely the spatio-temporal location data prediction. This type of services include, but not limited to, wireless traffic prediction, taxi supply and demand prediction, energy consumption prediction, air pollution prediction at specific locations. Data-driven, learning model-based solutions have demonstrated great data representation and generalization capability [11], [12], [13], [14].

However, the greatest difficulty that we confronted when applying machine learning models to localization and location data modeling lies in the big amount of labeled training data, which can be solved by aggregating small data collected from a large number of mobile users. Yet, such data gathering

Feng Yin, Zhidi Lin, Qinglei Kong and Shuguang (Robert) Cui are with both the Future Network of Intelligence Institute (FNii) and the School of Science and Engineering at The Chinese University of Hong Kong (Shenzhen) and Shenzhen Research Institute of Big Data (SRIBD), 518172, China. Email: yinfeng@cuhk.edu.cn, zhidilin@link.cuhk.edu.cn, kongqinglei@cuhk.edu.cn, shuguangcui@cuhk.edu.cn

Yue Xu was with Beijing University of Posts and telecommunications and is now with Alibaba corporation, Hangzhou, 311121, China. Email: avexuyue@gmail.com

Deshi Li is with the School of Electronic Information, Wuhan University, Wuhan, 430079, China. Email: dsli@whu.edu.cn

Sergios Theodoridis was with the Department of Informatics and Telecommunications at National and Kapodistrian University of Athens, 15784, Athens, Greece, and is now partially with The Chinese University of Hong Kong (Shenzhen) and SRIBD, Shenzhen, 518172, China. Email: stheodor@di.uoa.gr

*The corresponding author is Feng Yin. E-mail: yinfeng@cuhk.edu.cn

processes may cause severe data privacy issues, particularly when location is involved. As a special example, during the COVID-19 pandemic we have seen the value of sharing trajectories to track the spread of infections and predicting high-risk regions, meanwhile, there is an urgent need for location privacy preservation of the mobile users [15]. The federated learning framework is an outstanding solution for enhancing wireless localization accuracy and maintaining safe cooperation among users at the same time.

The gist of the proposed Federated Localization (FedLoc) framework is to let each mobile user/smart agent collect a smaller scale, local dataset and approximate the global machine learning model in a cooperative manner. Some concrete examples are as follows: (1) For static localization, a number of mobile users collect radio features at specific positions obtained either from the global positioning system (GPS) (for outdoor scenarios) or from the proximity to indoor reference points/landmarks (for indoor scenarios); (2) For target tracking and navigation, the mobile users collect diverse trajectories of inertial sensor- and wireless observations; (3) For wireless traffic prediction, base stations work as smart agents to collect local wireless data usage generated by their serving mobile users. We believe that the FedLoc framework is an up-and-coming solution for futuristic data-driven cooperative localization, not only because of the rapid development of distributed optimization techniques that serve as the algorithmic core, but also largely owing to the rapid development of smart phones with ever-increasing computation power and network throughput, the widespread use of quick-response (QR) codes, and the high-precision indoor/outdoor maps, altogether. Therefore, we believe it is timely to exploit all relevant federated learning techniques for localization and location data processing.

This overview paper is a four-mode mixture of review, new proposals, real evaluations, and outlook, being different from the majority that solely review the existing works. We focus on a specific application sector of federated learning, namely the data-driven cooperative localization and location data processing. The models and algorithms to be reviewed are carefully tailored for our desired applications. Besides, we focus on real use cases and their practical implementations from our own works as well as some other related works that all fall under this new cooperative paradigm. Detailed contributions of this overview paper are as follows.

- First, we propose a federated localization framework, called FedLoc, which elegantly addresses the privacy issue in cooperation among a massive number of mobile users for target localization and location data processing. We also proposed two potential wireless network infrastructures, namely a cloud-based one and an edge-based one, that can potentially help meet the communication requirements of the FedLoc framework.
- Second, we clarify the differences between the proposed FedLoc framework and the existing cooperative localization framework for sensor networks as well as the classic crowd-sourcing framework.
- Third, we review some state-of-the-art federated learning procedures, two widely used learning models, namely the deep neural network (DNN) and Gaussian process (GP), and a few distributed model hyper-parameter optimization schemes that work reasonably well for the two learning models. We put more emphasis on the Bayesian GP models than deterministic DNN models due to their unique welcome features for modeling location data.
- Fourth, we discuss four concrete use cases, namely (1) static target localization/fingerprinting; (2) outdoor vehicle navigation; (3) indoor pedestrian tracking; and (4) spatio-temporal wireless traffic prediction, to explain the use of the FedLoc framework. In the first use case, a static target localization system is built based on a DNN that maps a vector of radio features to a desired position. In the second use case, we propose a DNN-based accurate vehicle navigation with low-sampling-rate GPS. In the third case, the state transition function, as represented by the GP model, maps the current state to the next state in a non-parametric way for indoor pedestrian motion modeling. In the fourth use case, wireless traffic is modeled by a scalable GP under 5G Cloud-Radio Access Network (C-RAN) infrastructure. Various other related applications are also mentioned in this paper.
- Lastly, we evaluate the proposed FedLoc framework with real datasets for two aforementioned use cases to demonstrate their practical implementations and effectiveness in reality.

In this overview paper, we concentrate on federated learning tailored to target localization and location data processing. Due to the space limitation as well as the expertise of the authors, the following aspects are only briefly touched upon.

- Distributed optimization methods in the contexts of robustness, communication efficiency, and low-complexity. Some recent works include [16], [17].
- Adversarial attacks and advanced privacy-preserving schemes such as the block-chain based ones for federated learning. Some recent works include [18], [19], [20].
- General techniques and challenges of federated learning as well as its applications in other industry sectors, as surveyed by [21], [22].

The rest of this paper is organized as follows. In Section II, we briefly review the existing "cooperation" frameworks proposed primarily for wireless sensor networks. In Section III, we introduce two important learning models, namely the deep neural network and Gaussian process, for learning from data. In Section IV, we introduce the proposed FedLoc framework in detail, followed by two different wireless network infrastructures given in Section V to support the real deployment of the FedLoc framework. Various use cases of the proposed FedLoc framework are showcased in Section VI. Simulation results are given in Section VII to demonstrate the effectiveness of the FedLoc framework. In Section VIII, we discuss the major challenges of the FedLoc framework and give a few future research directions. Lastly, Section IX concludes this paper. Figure 1 gives a clear global picture of our work.

II. Related Work

In this section, we survey all related works and clarify their differences from our FedLoc framework to be introduced in

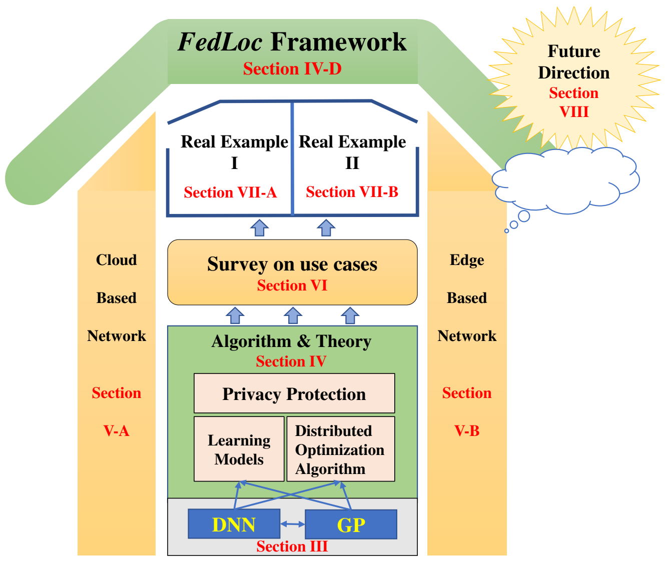


Fig. 1. Overall organization of this paper and links between different sections.

Section IV.

A. Sensor Network Localization

When speaking of “cooperation” in the context of wireless localization, it will certainly remind us the class of algorithms for determining a number of agents (nodes with unknown positions) with the aid of a few anchors (nodes with known positions) and a bunch of wireless measurements made between these nodes.

Cooperative localization has gained much attention since 2005 owing to the seminal work by Patwari and Hero [23], where they proposed to use the simple least-squares estimation criterion with time-of-arrival (ToA) or received-signal-strength (RSS) measurements to localize dozens of agents. The proposed method was evaluated with two sets of real measurements collected in an indoor environment. This seminal work has triggered a plethora of methods in the following years. Representative works include [24], [25], [26], [27], [28], [29], [30], to mention a few.

The fundamental differences between the aforementioned cooperative localization algorithms and our proposed FedLoc are the following:

- The aforementioned classic cooperative localization algorithms focus on determining unknown positions of a batch of distributed agents given their mutual position-related measurements. In this setting, position inference is the only task to be solved by the designed algorithm. In contrast, the proposed FedLoc aims to train a global learning model cooperatively by a batch of distributed devices with rather well-calibrated position-related data in the first place. After the global learning model has been trained, it can be used both by the existing distributed devices and new users to infer their positions with hopefully improved positioning accuracy.
- The above mentioned algorithms adopt empirical models, such as the log-distance path-loss model for RSS measurements [31], and Gaussian mixture model for non-line-of-sight propagation [29]. In contrast in the FedLoc framework, we solely consider data-driven, machine learning-based models.

B. Distributed Target Tracking

Distributed target tracking is mostly considered for sensor networks without a central node. For such network infrastructures, the traditional Kalman filter or particle filters cannot be used due to lack of the posterior belief/distribution of the desired target state (evolving in time) given all observed sensor measurements. To meet this challenge, various distributed implementations of the Kalman filter and particle filters, for instance [32], [33], [34], [35], were proposed with similar ideas of approximating the posterior belief/distribution as a product of local posteriors. Afterwards, local state estimates are communicated in a message consensus stage. The idea behind these two steps is similar to that of our FedLoc framework.

However, the major differences between the distributed target tracking and our FedLoc are the following:

- Distributed Kalman filter and particle filters are based on empirical models, while our FedLoc framework relies on data-driven, machine learning models.
- Distributed Kalman and particle filters exchange target state estimates directly over the air, which is fragile to malicious attacks; in contrast FedLoc trains a global deep learning model and advocates changing local model parameters under privacy-preserving schemes.
- Distributed Kalman and particle filters do not require training data, but need a good prior distribution of the initial target state. Therefore, they are agile for new deployments. In contrast, our FedLoc framework needs to train the global model beforehand.

C. Crowdsourcing

Crowdsourcing is a sourcing model in which services are built from a large, relatively open, and often rapidly-evolving group of internet users. Building and maintaining a location system/service based on crowdsourcing is somewhat related to our FedLoc idea. However, the state-of-the-art crowdsourcing methods place more emphasis on raw data sharing and aggregation from a bunch of collaborating users, therefore there is no model in mind. Representative works are as follows. In geography, voluntary users collaboratively build a street map, fill in street information, etc. OpenStreetMap (http://www.openstreetmap.com) and Wikimapia (http:// www.wikimapia.org) are two successful crowdsourcing projects among others. Crowdsourcing of virtual maps, such as RSS map or magnetic map, becomes trendy for big multi-storey buildings [36], [37], [38].

The fundamental differences between the crowdsourcing and the FedLoc are the following:

- Crowdsourcing is more about raw location data aggregation for map construction with less calibration effort, while position determination will be done in a separate stage later on. In contrast, FedLoc focuses on training

a global machine learning model for positioning in one step.
- Crowdsourcing is mostly model-free. In contrast, FedLoc is built around advanced machine learning models, making it diverse and vibrant.
- Crowdsourcing aggregates raw data without any safeguard, which will incur severe privacy issues. In contrast, FedLoc processes sensitive data locally and exchanges only the model hyper-parameters that are difficult to decode in general.

D. *Location Data Modeling*

In this paper, location data specifically refers to spatio-temporal data measured across space as well as time. Representative spatio-temporal data include environmental data, climate data, transportation data, human mobility data, social data, etc. Spatio-temporal data processing and modeling have been well studied over the past decades, ranging from traditional statistical methods to recent data-driven learning model-based methods. Traditional statistical methods include the autoregressive methods for multivariate random fields, factor analysis methods, stochastic process-based methods, tensor decomposition-based methods, see for instance [39], [40]. Data-driven learning models, such as recurrent neural network with long short-term memory and graph neural network have been used to model spatio-temporal data. A comprehensive survey on harnessing deep learning models for spatio-temporal data mining is given in [41]. A special note is given here on the Gaussian process model, which is also called Kriging in geostatistics and can be categorized into the traditional statistical models; however, it can also be regarded as a machine learning model for representing a spatial-temporal function with two inputs, namely the location and the time. In [42], [43], Gaussian processes implemented via recursive Kalman filtering are used to model spatio-temporal data with rather low computational complexity. Learning models are believed to be able to generate better modeling and prediction performance compared with the traditional statistical methods. In this work, we are keen on training learning models in a distributed manner by a large number of collaborating mobile users.

III. Learning Models

This section aims to introduce two representative learning models that can be used as the "brain" of the proposed FedLoc framework. We will first briefly review the deep neural network (DNN) model in Subsection III-A, followed by a short introduction to Gaussian process (GP) model in Subsection III-B. Lastly, we will shed some light on the connections of the two learning models and further highlight the benefits of using GP models over DNN models for FedLoc in Subsection III-C.

A. *Deep Neural Network*

Deep neural network (DNN) here refers to the class of feed-forward networks. The term "feed-forward" means data are fed from the input layer through several hidden layers to the output layer. Typically, a standard DNN, depicted in Fig.2, demonstrates a chain structure in math as

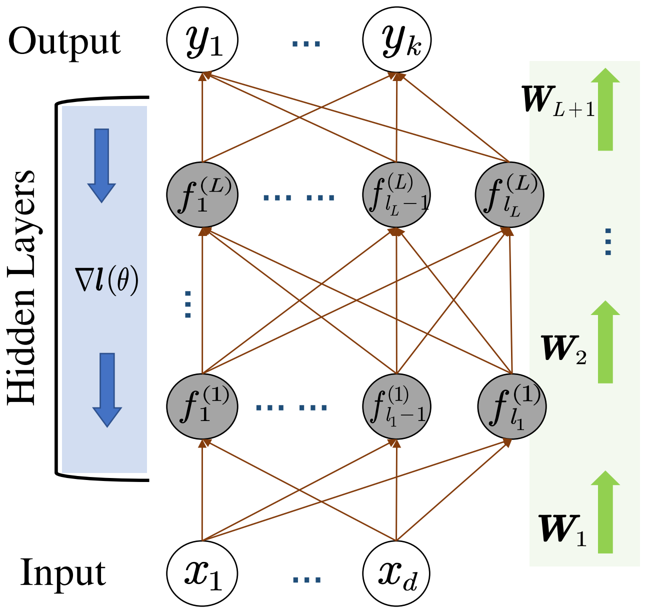


Fig. 2. Block diagram of deep neural network architecture. The input-, hidden-, and output variables are represented by nodes, and the weight parameters linking between the nodes at each layer are denoted by $\boldsymbol{W}_j$, where $j \in \{1, 2, \ldots, L+1\}$. $\boldsymbol{\theta} \triangleq \{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_{L+1}\}$ comprises all model hyper-parameters, namely the neural network weights of all layers. Green arrows indicate the forward direction of information flow through the network in the inference stage, while the blue arrows indicate the backward direction of the gradient flow for hyper-parameter optimization using back-propagation by default.

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{W}_{L+1} f^{(L)}\left(\cdots \boldsymbol{W}_3 f^{(2)}\left(\boldsymbol{W}_2 f^{(1)}(\boldsymbol{W}_1 \boldsymbol{x})\right)\right), \tag{1}$$

starting from the inputs/features $\boldsymbol{x}$ and passing L hidden layers to the output. Here we term $\boldsymbol{W}_j$ the weight matrix in the j-th layer where $j \in \{1, 2, \ldots, L+1\}$ and $\boldsymbol{\theta}$ the collection of all hyper-parameters, i.e., $\boldsymbol{\theta} \triangleq \{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_{L+1}\}$. The mapping function $f^{(j-1)}(\cdot)$[1] in each hidden layer, comprises a bunch of elementary activation functions that mimic the role of neurons in our brain. The commonly used activation functions include the sigmoid function, rectified linear unit (ReLU) function, and some other variants. According to the universal approximation theorem [44], a DNN can well approximate any smooth function by tuning the number of hidden layers and the number of neurons in each hidden layer.

Given n training samples $\mathcal{D} \triangleq \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, one can train the model hyper-parameters $\boldsymbol{\theta}$, such that the network output $f(\boldsymbol{x}; \boldsymbol{\theta})$ is close to the ground truth. Often, DNNs are trained through minimizing the difference between $f(\boldsymbol{x}; \boldsymbol{\theta})$ and y. The minimization problem for a set of n training samples can be written as

$$\min_{\boldsymbol{\theta}} \; l(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})), \tag{2}$$

where $\ell(\cdot, \cdot)$ is a certain loss function, e.g., the quadratic loss function. In this paper, the input $\boldsymbol{x}_i$ represents position related

[1] $f^{(0)}(\boldsymbol{x}) = \boldsymbol{x}$

Optimization Method	Update Formulations	Pros	Cons
Gradient Descent (GD)	$\boldsymbol{\theta}^{\eta+1} = \boldsymbol{\theta}^{\eta} - \gamma_t \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \mathcal{D})$	• converge to global minimum for convex loss surface	• slow convergence and intractable for large dataset • not suitable for online applications and easily getting trapped in saddle points when optimizing non-convex loss surface • choosing learning rate difficultly
Stochastic Gradient Descent (SGD)	$\boldsymbol{\theta}^{\eta+1} = \boldsymbol{\theta}^{\eta} - \gamma_t \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \boldsymbol{x}_i, y_i)$	• faster convergence • suitable for online applications	• high fluctuation of the gradients and outcomes • easily getting trapped in saddle points • choosing learning rate difficultly
Mini-batch GD	$\mathcal{D}_m \triangleq$ traning batch $\boldsymbol{\theta}^{\eta+1} = \boldsymbol{\theta}^{\eta} - \gamma_t \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \mathcal{D}_m)$	• mediate convergence • smaller updates fluctuations • more stable convergence compared to SGD	• easy to get trapped in saddle points • difficult to choose a proper learning rate
Adaptive gradient methods: SGD with Nesterov momentum [45], AdaGrad [46], RMSProp [47], Adam [48] etc.	/	• boost robustness of SGD • suitable for dealing with sparse data • less manual tuning of the learning rate	• might cause infinitesimally small learning rate

TABLE I
COMMONLY USED NEURAL NETWORK TRAINING ALGORITHMS

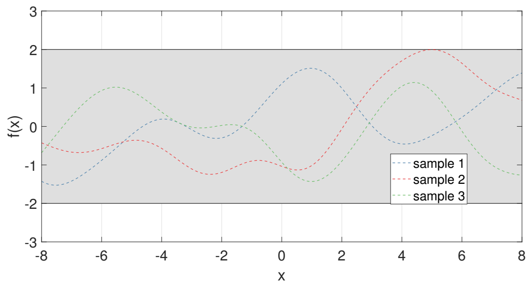
measurements, and y_i represents positions most of the time. We will provide some concrete examples in Section IV, V and VI. Gradient descent type methods with back-propagation are commonly used to solve the above minimization problem in spite of its numerical instability caused by gradient vanishing or explosion. After the optimal set of weights $\hat{\boldsymbol{\theta}}$ is obtained, one can conduct prediction for a novel input $\boldsymbol{x}_*$ using $f(\boldsymbol{x}_*; \hat{\boldsymbol{\theta}})$ given in Eq.(1).

It is worth mentioning at this point that after decades of exploration, people have summarized various useful tricks to train large networks effectively. For instance, careful initialization of DNN weights and proper normalization are effective to resolving the headache from gradient explosion/vanishing. Widely used initialization techniques include LeCun initialization [49], Xavier initialization [50], Kaiming initialization [51], and so on, while normalization techniques include batch normalization [52], weight normalization [53], layer normalization [54], and so on. For more training tricks, interested readers may refer to [55], [56], [57], [58] for a better structured tutorial, and next we will turn to a brief review of state-of-the-art optimization algorithms for DNN training.

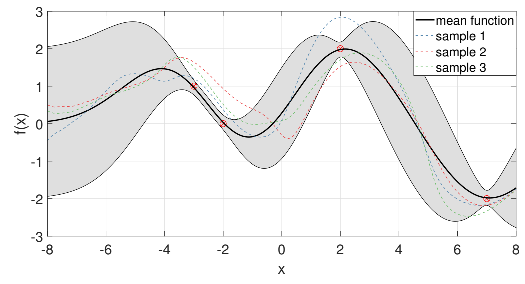
Commonly used DNN training methods are presented in Table I, wherein η represents the iteration index. According to the amount of data used in computing the gradient $\nabla_{\boldsymbol{\theta}} \ell$, there are generally three types of gradient descent methods, namely batch gradient descent (GD), stochastic gradient descent (SGD) and mini-batch gradient descent. Their corresponding update formulations, pros and cons are shown in Table I. However, these classic methods all face the difficulty of manually selecting the learning rate γ_t. More recently, people turn to adopting modern adaptive gradient methods such as RMSProp [47] and Adam [48] to address this issue. It is said that such adaptive gradient methods are likely to achieve the best performance when the input data is sparse [55]. However, this does not necessarily mean the adaptive gradient methods are always superior to other type of GD-based methods. For more details about the different optimization algorithms, interested readers can refer to [55], [56] and the reference therein.

The DNN structure has a big impact on both the forward-propagation and back-propagation computational complexity. For ease of exposition, a specific DNN structure is depicted in Fig. 2, wherein we assume L hidden layers and n neurons in each hidden layer, being of the same order as the data size. Typically, we assume $n \gg L$. Moreover, we assume the number of data samples n is way larger than the feature dimension d, i.e., $n \gg d$. For this configuration, the computational complexity required by the forward-propagation is mainly due to the product of the weight matrix and the input vector in each layer, namely, $\boldsymbol{W}_j f^{(j-1)}(\boldsymbol{x})$, where $j \in \{1, 2, \ldots, L+1\}$, thus scales as $\mathcal{O}(n^2)$ for one single data sample. The overall computational complexity of the forward-propagation is $\mathcal{O}(n^3)$ for n data samples. As to the back-propagation, let us first note that evaluating $l(\boldsymbol{\theta})$ in each iteration of the gradient descent step requires a forward propagation. Assuming that the gradient descent runs k ($k \ll n$) iterations, the computational complexity for the back-propagation scales as $\mathcal{O}(n^3)$ too.

The aforementioned DNN is suitable for tabular data in general. However, there exist a plethora of deep variants for data with unique features, such as convolutional networks [59] and capsule networks [60] for images, long-short-term-memory (LSTM) networks for sequential data, and graph neural networks [61] for spatial and spatio-temporal data. In order to reduce the size of a deep model as well as its computational complexity for use on smartphone and edge devices, one could resort to model distillation techniques [62] or model sparsification techniques [63].



(a) Prior



(b) Posterior

Fig. 3. Subfigure (a) shows three sample functions drawn randomly from a GP prior with a specific squared-exponential kernel. Subfigure (b) shows three sample functions drawn from the posterior conditioned on the prior in (a) as well as four noisy observations indicated by red dots. The corresponding posterior mean function is depicted by the black curve. The grey shaded area represents the uncertainty region, namely the 95% confidence region for both the prior and the posterior, respectively.

B. Gaussian Processes

Gaussian processes (GP) constitute an important class of Bayesian non-parametric models, which are closely related to several other salient machine learning models. A Gaussian process is a collection of random variables, any finite subset of which follows a Gaussian distribution [1]. In the sequel, we solely focus on scalar, real-valued Gaussian processes that are completely specified by a mean function and a kernel function (a.k.a. covariance function). Concretely,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_h)), \tag{3}$$

where $m(\boldsymbol{x})$ is the mean function, which is often set to zero in practice, especially when there is no prior knowledge about the underlying process; and $k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_h)$ is the kernel function tuned by the kernel hyper-parameters, $\boldsymbol{\theta}_h$.

Let us consider the GP regression model, $y = f(\boldsymbol{x}) + e$, where $y \in \mathbb{R}$ is a continuous-valued, scalar output; the unknown function $f(\boldsymbol{x}) : \mathbb{R}^d \mapsto \mathbb{R}$ is modeled as a zero mean GP; and the noise e is assumed to be Gaussian distributed with zero mean and variance σ_e^2. Moreover, the noise terms at different data points are assumed to be mutually independent. The set of all unknown GP hyper-parameters is denoted by $\boldsymbol{\theta} \triangleq [\boldsymbol{\theta}_h^T, \sigma_e^2]^T$, and the dimension of $\boldsymbol{\theta}$ is assumed to be equal to p.

Based upon these GP regression model settings, given training dataset $\mathcal{D} \triangleq \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ and test dataset $\mathcal{D}_* \triangleq \{\boldsymbol{x}_{*i}, y_{*i}\}_{i=1}^{n_*}$, the joint prior distribution of the training output $\boldsymbol{y}$ and test output $\boldsymbol{y}_*$ can be written compactly as

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_e^2 \boldsymbol{I}_n, & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}), & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) + \sigma_e^2 \boldsymbol{I}_{n_*} \end{bmatrix}\right), \tag{4}$$

where $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ is an $n \times n$ covariance matrix between the training inputs; $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)$ is an $n \times n_*$ covariance matrix between the training inputs and test inputs, $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*)$ is an $n_* \times n_*$ covariance matrix between the test inputs. Here, we let $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ be the short term of $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_h)$.

Applying some known results of conditional Gaussian distribution, we can easily derive the posterior distribution as

$$p(\boldsymbol{y}_* | \mathcal{D}, \boldsymbol{X}_*; \boldsymbol{\theta}_h) \sim \mathcal{N}\left(\bar{\boldsymbol{m}}, \bar{\boldsymbol{V}}\right), \tag{5}$$

where the posterior mean (vector) and the posterior covariance (matrix) are respectively,

$$\begin{aligned} \bar{\boldsymbol{m}} &= \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_e^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y}, & (6) \\ \bar{\boldsymbol{V}} &= \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) + \sigma_e^2 \boldsymbol{I}_{n_*} \\ &\quad - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_e^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*). & (7) \end{aligned}$$

Given a novel input in the test dataset, the above posterior mean gives the prediction, while the posterior covariance gives the uncertainty region of the prediction. A graphical illustration of GP working on a toy regression example is showcased in Fig. 3

Just like the choice of activation functions and architecture that highly affect the performance of neural networks, the kernel function determines the expressive power of the GP model to a large extent. More specifically, the kernel function profoundly controls the characteristics (e.g., smoothness and periodicity) of a family of functions. Therefore, in order to make a kernel function full of expressive power and automatically adaptive to a given dataset, the following works can be adopted. In [64], a spectral mixture (SM) kernel was proposed to approximate the spectral density with a Gaussian mixture model arbitrarily well in the frequency domain and transform it back into a universal stationary kernel. In [65], [66], the authors modified the SM kernel to a linear multiple low-rank sub-kernels with a favorable optimization structure, which enables faster and more stable numerical search. In [67], [68], [69], [70], a DNN architecture was combined with the automatic relevance determination (ARD) kernel to approximate any kernel function (including both the stationary and non-stationary ones). Yet, in a more recent trend designing universal kernels may be obtained as a byproduct of designing new fashioned deep GP models [71] that link DNNs to GPs [72], [73], [74].

Next, we introduce the classical ML-based GP hyper-parameter estimation. Due to the Gaussian assumption on the noise, the log-likelihood function can be obtained in closed form. The GP hyper-parameters can be optimized equivalently by minimizing the negative log-likelihood function:

$$l(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) = \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\theta}) \boldsymbol{y} + \log\det\left(\boldsymbol{C}(\boldsymbol{\theta})\right), \tag{8}$$

where $\boldsymbol{C}(\boldsymbol{\theta}) \triangleq \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_h) + \sigma_e^2 \boldsymbol{I}_n$ and $\det(\cdot)$ denotes the determinant of a matrix. This optimization problem is mostly solved via gradient descent type methods, such as LFGS-Newton or conjugate gradient [1], which requires the following closed-form partial derivatives, for $i = 1, 2, ..., p$,

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_i} = tr\left(\boldsymbol{C}^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}(\boldsymbol{\theta})}{\partial \theta_i}\right) - \boldsymbol{y}^T\boldsymbol{C}^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}(\boldsymbol{\theta})}{\partial \theta_i}\boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y}, \tag{9}$$

where $tr(\cdot)$ denotes the trace of a square matrix.

It should be noted that the minimization problem in Eq.(8) may easily get stuck at a bad local optimum when the selected learning model is over-parameterized and the associated cost function does not show any favorable optimization structure.

Using the above ML method to train a GP model requires $O(n^3)$ computational complexity thus forbids its practical use. To address this difficulty, a plethora of scalable GP models have been developed in the past decades. Some representative works of different categories were obtained through using (1) low-rank kernel matrix approximation [75]; (2) local structures of the kernel matrix [76]; (3) the state-space model reformulation and Kalman filter [42]; (4) the Bayesian committee machine (BCM) with a number of distributed computing units [77]; and (5) the variational Bayesian formulation [78]. A comprehensive survey of the existing scalable GP models can be found in [79].

C. DNN Versus GP

In the previous subsections, we briefly introduced DNN and GP that can both be used as the core learning model. DNN is quite popular nowadays due to various good reasons. Among others, it can approximate any smooth function according to the universal approximation theorem [44]. But the main drawbacks of DNN lie in its opaque model interpretability and the large number of hyper-parameters (DNN weights) to be trained. For our FedLoc framework proposed in this paper, we put more emphasis on the GP models due to their unique welcome features compared with DNN.

First, GP models involve significantly fewer model hyper-parameters than an equally-effective DNN. From [80] we know that a single layer Bayesian neural network with i.i.d. weights converges to a GP. Consequently, a neural network kernel was designed with the following explicit form [1]:

$$k_{\text{NN}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\boldsymbol{x}}\Sigma\tilde{\boldsymbol{x}}'}{\sqrt{(1+2\tilde{\boldsymbol{x}}\Sigma\tilde{\boldsymbol{x}})(1+2\tilde{\boldsymbol{x}}'\Sigma\tilde{\boldsymbol{x}}')}}\right), \tag{10}$$

where $\tilde{\boldsymbol{x}} \triangleq [1, \boldsymbol{x}^T]^T$ is an augmented input vector. Often, we assume $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_{d+1}^2)$ to be a diagonal matrix, thus the hyper-parameters $\boldsymbol{\theta}_h = [\sigma_1^2, \sigma_2^2, ..., \sigma_{d+1}^2]^T$ is of dimension $d+1$. If Σ is taken to be a general matrix, the hyper-parameters to be tuned is in the order of d^2, being much smaller than the size of a fully-connected DNN in general.

Lately, the arc-cosine kernel [72], the neural tangent kernel (NTK) [81], and the convolutional neural tangent kernel (CNTK) [82] were developed to mimic a DNN with infinite width. The arc-cosine kernel function [72] is given by

$$k_{\text{arccos}}(\boldsymbol{x}, \boldsymbol{x'}) = 2\int \frac{e^{-\frac{\|\mathbf{w}\|^2}{2}}}{(2\pi)^{d/2}}\Theta(\mathbf{w}\cdot\boldsymbol{x})\Theta(\mathbf{w}\cdot\boldsymbol{x'})(\mathbf{w}\cdot\boldsymbol{x})^q(\mathbf{w}\cdot\boldsymbol{x'})^q d\mathbf{w}, \tag{11}$$

where $\Theta(z) = \frac{1}{2}(1+\text{sign}(z))$ denotes the Heaviside step function, and q is a non-negative integer for selecting a particular activation function. The arc-cosine kernel for multi-layer neural network can also be obtained via a recursive kernel design. The hyper-parameters of the arc-cosine kernel include the kernel order parameter q for specifing the activation function and the number of hidden layers L.

The NTK captures the behavior of fully-connected deep neural networks trained by gradient descent, and CNTK is an extension of NTK to convolutional neural networks. The analytic form of NTK can be derived recursively as

$$k_{\text{NTK}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h=1}^{L+1}\left(\Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \prod_{h'=h}^{L+1}\dot{\Sigma}^{(h')}(\boldsymbol{x}, \boldsymbol{x}')\right), \tag{12}$$

where $\Sigma^{(h-1)}$ is the centered covariance matrix of the $(h-1)$th layer's output $f^{(h)}(\boldsymbol{x})$, and $\dot{\Sigma}$ is the corresponding derivative covariance.

It can be proven that a sufficiently wide and randomly initialized DNN trained by gradient descent is equivalent to a kernel regression predictor with the aforementioned NTK kernel. Hence, the properties of the ultra-wide DNN, such as the generalization capability, can be obtained by learning the corresponding NTK, albeit with much less computational effort. It is also noteworthy that the hyper-parameter of the NTK is only the number of layers that can be tuned easily using cross-validation.

Second, GP models can handle input uncertainty naturally. In our considered applications, the model inputs often involve position or position related measures that are intrinsically subject to noise due to imperfect field calibration. Since GP model is a probabilistic model, the input uncertainty can be easily incorporated into the model. One way is to assume the training input $\boldsymbol{x}$ to be a random variable with a known distribution $p(\boldsymbol{x})$. In [83], for instance, the mean function of GP with input uncertainty was obtained as

$$\tilde{m}(\boldsymbol{x}) = \int m(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \tag{13}$$

and the kernel function obtained as

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \iint k(\boldsymbol{x}, \boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}'. \tag{14}$$

The only difficulty lies in the evaluation of the two integrals. In general, they can be approximated by Monte-Carlo integration [57], [84]. The rest of the steps remain the same as the standard GP with clean input as given in (1). The computation can be largely reduced for Gaussian distributed input $\boldsymbol{x}$ using unscented transform, see for instance [85, Chapter 5.5].

Third, GP models can more easily encode prior information about the data than DNN. This is inherited from the meaningful interpretation of various elementary kernels with known characteristics. For instance, when the data demonstrate periodicity, we could add elementary periodic kernel(s) or locally periodic kernel(s) to the eventual composite kernel; when the data demonstrate linear rising trend, we could add a linear kernel to the eventual kernel; when the data profile is verified to be smooth, we could use the squared-exponential (SE) kernel with a large length scale parameter. Taking into account the prior information about the data can be regarded as regularizing the model fitting process, thus is effective for avoiding data over-fitting. This is a welcome feature for our applications in which the total amount of data is large but each mobile user may only have a small amount of local data in hand for training the global model. According to a recent white paper released by Huawei, wireless big data in 6G will be generated by a huge amount of mobile users and IoT devices, each contributing only a small local dataset.

Fourth, GP models own a more robust generalization capability compared to DNNs. Recently, double descent phenomenon [86] has once again attracted broad attention. Such phenomenon describes that with the increase of model complexity, generalization error first decreases, increases, and then again decreases, which has been presented as mysterious generalization behavior in various learning algorithms [87]. Recent work in [88] further showed that double descent in DNNs occurs not just as a function of model complexity, but also as a function of the number of training samples, hence it can lead to a regime where training with more data leads to worse test performance. By contrast, Bayesian models is more robust in terms of generalization capability due to the effect of marginalization. Specifically, instead of using the point estimation of the DNN models, Bayesian learning models make the prediction in a different way, i.e.,

$$p(y_* \mid \boldsymbol{x}_*, \mathcal{D}) = \int p(y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) d\boldsymbol{\theta}. \tag{15}$$

That is, instead of taking only one single setting of parameters $\boldsymbol{\theta}$, Bayesian learning models consider the whole parameter space, weighted by the posterior probabilities, and finally marginalize out $\boldsymbol{\theta}$ to obtain the eventual predictive distribution. This procedure can be interpreted as model averaging, and it has been shown that such Bayesian model averaging can alleviate even prominent double descent behavior [89]. If a reasonable prior $p(\boldsymbol{\theta})$ is selected, the regression performance is supposed to monotonically improve as we increase the training sample size or the model complexity. This suggests that GP models, as an important class of Bayesian non-parametric models, are insensitive to double descent phenomenon.

Finally, it is noteworthy that DNNs and their variants are still more widely used than GPs for machine learning empowered applications. But for localization applications, yet, GP models are very promising due to the aforementioned advantages.

IV. Federated Localization (FEDLOC)

The organization of this section is the following. In Subsection IV-A, the main idea of federated learning is introduced, followed by a review of various existing distributed training methods proposed for DNN and GP learning models in Subsection IV-B. Privacy-preserving schemes are briefly surveyed in Subsection IV-C. Lastly, we conclude this section by giving a full picture of the FedLoc framework.

A. Brief Review of Federated Learning

The idea of federated learning exists for a long time in the context of distributed learning, and it was given the name by some researchers at Google in 2016 [90], [2]. Federated learning is a flexible and safe cooperation framework for mobile users. The idea behind the federated learning is to approximate a global model/objective as a summation of local models/objectives trained individually by mobile users. Mathematically, the above idea can be expressed as

$$l(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) \approx \sum_{k=1}^{K} l^{(k)}(\boldsymbol{X}_k, \boldsymbol{y}_k; \boldsymbol{\theta}), \tag{16}$$

where $\boldsymbol{X}$ is the complete set of the training inputs, $\boldsymbol{y}$ is the complete set of the training outputs, and they constitute the complete training set $\mathcal{D}$; $l(\cdot)$ is a global objective in terms of the model hyper-parameters $\boldsymbol{\theta}$; while $\boldsymbol{X}_k$ is the k-th local set of the training inputs, $\boldsymbol{y}_k$ is the k-th local set of the training outputs, and they constitute $\mathcal{D}_k$, which is a subset of $\mathcal{D}$; $l^{(k)}(\cdot)$ is a local objective of the k-th local dataset, $\mathcal{D}_k$; K is the total number of collaborating mobile users, which is assumed to be large. Both $l(\cdot)$ and $l^{(k)}(\cdot)$ are composite functions of a selected learning model/regression function and a cost function. Lastly, we note that the outputs $\boldsymbol{y}$ are mostly positions or position related measurements in our work.

To shed some light on the objective $l(\cdot)$, let us consider the following two different machine learning models and their cost functions.

I: DNN model with the Least-Squares Cost. The global objective for training a DNN is given as follows:

$$l(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta})\right)^2, \tag{17}$$

where the outputs are assumed to be independent, and $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ is represented by a DNN with L hidden layers [58] with $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, ..., \boldsymbol{W}_{L+1}\}$ representing the DNN weights to be tuned for all hidden layers and output layer. It is obvious that the global objective is already in form of sum-of-residual-squared.

II: GP model with the Maximum Likelihood Cost. Due to the Gaussian assumption on the noise, described in Subsection III-B, the log-likelihood function can be obtained in closed form. Therefore, the global objective for training the GP regression model hyper-parameters is

$$\begin{aligned} l(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) &= \log p(\boldsymbol{y}; \boldsymbol{X}, \boldsymbol{\theta}) \\ &= \log \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{m}(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta})\right), \end{aligned} \tag{18}$$

where the vector $\boldsymbol{m}(\boldsymbol{X})$ and the matrix $K(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta})$ are respectively the mean function $m(\boldsymbol{x})$ and the kernel function $k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$ evaluated for the complete dataset $\mathcal{D}$. This global objective is not directly in the form of summation, but commonly approximated by the product-of-expert (PoE) [77] as

$$l(\boldsymbol{X},\boldsymbol{y};\boldsymbol{\theta}) \approx \sum_{i=1}^{K} \log \mathcal{N}\left(\boldsymbol{y}_k; \boldsymbol{m}(\boldsymbol{X}_k), K(\boldsymbol{X}_k,\boldsymbol{X}_k;\boldsymbol{\theta})\right). \tag{19}$$

Here, we note that the independent noise term has been absorbed into the kernel function for notation brevity in Eq.(18) and Eq.(19).

B. Distributed Training of the Learning Models

The original goal is to train a global model through

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} l(\boldsymbol{X},\boldsymbol{y};\boldsymbol{\theta}), \tag{20}$$

where the objective function is often non-convex and solved by gradient descent type methods. When the complete dataset is large, training the global model given above can be computationally expensive. As mentioned before, federated learning aims to distribute the heavy computation load to a massive number of collaborating mobile users by solving instead the following problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} l^{(k)}(\boldsymbol{X}_k,\boldsymbol{y}_k;\boldsymbol{\theta}). \tag{21}$$

Each mobile user maintains a local update of the global model hyper-parameters and sends it to a central node for consensus. There exist various ways for updating the global model hyper-parameters. In the following, we introduce the classical federated averaging (FedAvg) [2] algorithm and a few algorithms developed upon alternating direction of multipliers method (ADMM) [91], [92].

We start with the state-of-the-art FedAvg algorithm. Typically, the k-th mobile user calculates the gradient $\nabla l^{(k)}(\boldsymbol{\theta})$ and uploads it to the central node. The central node then aggregates a batch of/all local gradients to approximate $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{X},\boldsymbol{y};\boldsymbol{\theta})$. We illustrate this workflow in Fig. 4(a), which is named by FedAvg and deemed as the optimization algorithmic core of the federated learning framework [2]. A robust variant, called FedProx [93], was proposed to improve local training convergence by adding an extra proximal step at each client to restrict the distance between the local parameter estimates and the current global estimate.

Next, we introduce two ADMM-based hyper-parameter optimization schemes, that can effectively balance the computation and communication efficiency. The first one, namely the classical ADMM-based hyper-parameter optimization scheme (short as cADMM), reformulates the optimization problem in (21) as a nonconvex consensus problem [91] with a set of newly introduced local hyper-parameters $\{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_K\}$ and the global hyper-parameter $\boldsymbol{z}$. Concretely, we solve instead

$$\begin{aligned} \min \quad & \textstyle\sum_{k=1}^{K} l^{(k)}(\boldsymbol{\theta}_k), \\ \text{s.t.} \quad & \boldsymbol{\theta}_k - \boldsymbol{z} = \boldsymbol{0}, \quad \forall\, k=1,2,\ldots,K, \end{aligned} \tag{22}$$

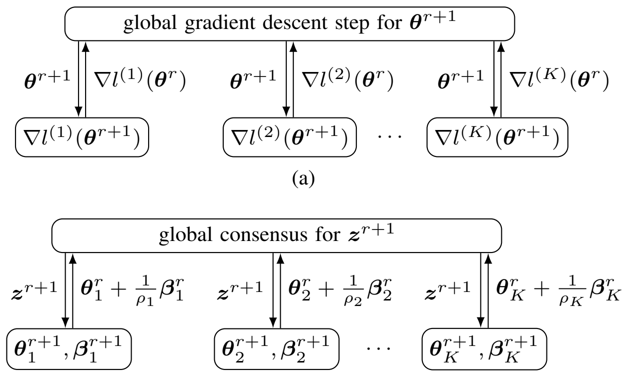

global gradient descent step for $\boldsymbol{\theta}^{r+1}$
$\boldsymbol{\theta}^{r+1}$
$\nabla l^{(1)}(\boldsymbol{\theta}^{r})$
$\nabla l^{(2)}(\boldsymbol{\theta}^{r})$
$\nabla l^{(K)}(\boldsymbol{\theta}^{r})$
$\nabla l^{(1)}(\boldsymbol{\theta}^{r+1})$
$\nabla l^{(2)}(\boldsymbol{\theta}^{r+1})$
$\cdots$
$\nabla l^{(K)}(\boldsymbol{\theta}^{r+1})$
(a)
global consensus for $\boldsymbol{z}^{r+1}$
$\boldsymbol{z}^{r+1}$
$\boldsymbol{\theta}_1^r + \frac{1}{\rho_1}\boldsymbol{\beta}_1^r$
$\boldsymbol{\theta}_2^r + \frac{1}{\rho_2}\boldsymbol{\beta}_2^r$
$\boldsymbol{\theta}_K^r + \frac{1}{\rho_K}\boldsymbol{\beta}_K^r$
$\boldsymbol{\theta}_1^{r+1}, \boldsymbol{\beta}_1^{r+1}$
$\boldsymbol{\theta}_2^{r+1}, \boldsymbol{\beta}_2^{r+1}$
$\cdots$
$\boldsymbol{\theta}_K^{r+1}, \boldsymbol{\beta}_K^{r+1}$
(b)

Fig. 4. Workflow of two existing distributed hyper-parameter optimization schemes. (a) FedAvg [2]. (b) cADMM [91].

where $l^{(k)}(\boldsymbol{\theta}_k)$ is nonconvex in terms of the local hyper-parameter $\boldsymbol{\theta}_k$ in general. The augmented Lagrangian function for Eq.(22) is given by

$$\begin{aligned} \mathcal{L}(\{\boldsymbol{\theta}_k\},\boldsymbol{z},\{\boldsymbol{\beta}_k\}) \;=\; & \textstyle\sum_{k=1}^{K}(l^{(k)}(\boldsymbol{\theta}_k) + \boldsymbol{\beta}_k^T(\boldsymbol{\theta}_k - \boldsymbol{z}) \\ & + (\rho_k/2)\|\boldsymbol{\theta}_k - \boldsymbol{z}\|_2^2), \end{aligned} \tag{23}$$

where $\boldsymbol{\beta}_k$ is a dual variable, and ρ_k stands for a predetermined regularization parameter. The $(r+1)$-th iteration of the cADMM for solving (Eq.22) can be decomposed into

$$\begin{aligned} \boldsymbol{z}^{r+1} &= \frac{1}{K}\textstyle\sum_{k=1}^{K}(\boldsymbol{\theta}_k^r + \frac{1}{\rho_k}\boldsymbol{\beta}_k^r), && \text{(24a)} \\ \boldsymbol{\theta}_k^{r+1} &= \arg\min_{\boldsymbol{\theta}_k}(l^{(k)}(\boldsymbol{\theta}_k) + (\boldsymbol{\beta}_k^r)^T(\boldsymbol{\theta}_k - \boldsymbol{z}^{r+1}) \\ &\quad + (\frac{\rho_k}{2})\|\boldsymbol{\theta}_k - \boldsymbol{z}^{r+1}\|_2^2), && \text{(24b)} \\ \boldsymbol{\beta}_k^{r+1} &= \boldsymbol{\beta}_k^r + \rho_k(\boldsymbol{\theta}_k^{r+1} - \boldsymbol{z}^{r+1}). && \text{(24c)} \end{aligned}$$

The above workflow is shown in Fig. 4(b) for clarity.

Next, we continue to introduce a more recent proximal ADMM (short as pxADMM) scheme proposed in [92], which is capable of reducing the communication overhead and the computational time *at the same time*. Unlike in step Eq.(24b) where the local hyper-parameters $\boldsymbol{\theta}_k$ are updated through minimizing the augmented Lagrangian function exactly, the proximal ADMM takes a proximal step w.r.t. $\boldsymbol{\theta}_k$ by applying the first-order Taylor expansion to $l^{(k)}(\boldsymbol{\theta}_k)$ [92], i.e.,

$$\begin{aligned} \boldsymbol{\theta}_k^{r+1} = & \arg\min_{\boldsymbol{\theta}_k} \nabla^T l^{(k)}(\boldsymbol{z}^{r+1})(\boldsymbol{\theta}_k - \boldsymbol{z}^{r+1}) \\ & + (\boldsymbol{\beta}_k^r)^T(\boldsymbol{\theta}_k - \boldsymbol{z}^{r+1}) + \left(\frac{\rho_k + L_k}{2}\right)\|\boldsymbol{\theta}_k - \boldsymbol{z}^{r+1}\|_2^2, \end{aligned} \tag{25}$$

where L_k is a newly introduced positive constant making $\|\nabla l^{(k)}(\boldsymbol{\theta}_k) - \nabla l^{(k)}(\boldsymbol{\theta}'_k)\| \le L_k\|\boldsymbol{\theta}_k - \boldsymbol{\theta}'_k\|$ satisfied for all $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}'_k, k=1,2,\ldots,K$. Note that the proximal step in Eq.(25) for $\boldsymbol{\theta}_k$ is a (convex) quadratic optimization problem with the following closed-form solution:

$$\boldsymbol{\theta}_k^{r+1} \;=\; \boldsymbol{z}^{r+1} - \left(\frac{\nabla l^{(k)}(\boldsymbol{z}^{r+1}) + \boldsymbol{\beta}_k^r}{\rho_k + L_k}\right). \tag{26}$$

As a consequence, the $(r+1)$-th iteration of the pxADMM for solving Eq.(22) can be decomposed into

$$\boldsymbol{z}^{r+1} = (1/K)\textstyle\sum_{k=1}^{K}(\boldsymbol{\theta}_k^r + \frac{1}{\rho_k}\boldsymbol{\beta}_k^r), \tag{27a}$$

$$\boldsymbol{\theta}_k^{r+1} = \boldsymbol{z}^{r+1} - \frac{(\nabla l^{(k)}(\boldsymbol{z}^{r+1}) + \boldsymbol{\beta}_k^r)}{\rho_k + L_k}, \tag{27b}$$

$$\boldsymbol{\beta}_k^{r+1} = \boldsymbol{\beta}_k^r + \rho_k(\boldsymbol{\theta}_k^{r+1} - \boldsymbol{z}^{r+1}). \tag{27c}$$

The pxADMM shares the same workflow with the cADMM as depicted in Fig. 4(b). Criteria for choosing ρ_k and L_k are given in [92], where the authors also proved under mild conditions that: (1) $\boldsymbol{\theta}_k^r$ converges to $\boldsymbol{z}^r$ for all k; and (2) solution $(\{\boldsymbol{\theta}_k^r\}, \boldsymbol{z}^r, \{\boldsymbol{\beta}_k^r\})$ converges to a stationary point of Eq.(22).

The pxADMM reduces the communication overhead in the same way as cADMM does, which was explained in our previous work [11]. However, the proximal step shown in Eq.(27b) leads to an inexact, but closed-form solution of the local sub-problem Eq.(24b) with much cheaper computation cost. Although more iterations may be required towards convergence, the overall computational time can be well reduced.

Yet, we must point out that there exist various different alternatives for distributed optimization, such as gossip algorithms [94], as well as some new alternatives including for instance distributed second-order Newton methods [95], distributed zero-order optimization methods [96] and parallel coordinate descent [97], etc. Depending on the task specification, these alternatives might be more effective than ADMM.

C. Privacy Preservation

Federated learning emphasizes strongly on mobile users sole ownership of data and preservation of user privacy. However, recent studies have shown that the shared parameters of the trained models are proved to be vulnerable to disclose sensitive information [98]. Privacy preservation in federated learning can be achieved through various security techniques like secure multi-party computation, homomorphic encryption, and differential privacy.

To protect the content of each individual piece of trained model, secure multi-party computation involves multiple participants to upload trained models towards the server collaboratively. No matter DNN or GP is used, the distributed gradient descent on user-held training data is protected by secure aggregation with user dropout taken into consideration [99]. By exploiting a secure aggregation protocol and a secret-sharing scheme, the privacy of each user-provided model can be guaranteed under an honest-but-curious and active adversarial setting [100], which supports an arbitrary subset of user dropouts. Other than the above schemes, to verify the correctness of the final aggregation result, a privacy-preserving and verifiable federated learning protocol has been designed with a homomorphic hash function and a secret sharing protocol [101]. However, secure multi-party computation may still leak sensitive information during the learning process.

The key idea of differential privacy in federated learning is to add some noise to the trained hyper-parameters with a sensitivity-measured random mechanism, such as Laplace mechanism or Gaussian mechanism [102], which helps mitigate the risk of private information disclosure. However, the injected noise may degrade the performance of the trained model. The feasibility of differential privacy on a client level in federated learning with Gaussian mechanism was demonstrated in [103], in which the authors demonstrated the trade-off between the loss of privacy and the modeling performance.

Various homomorphic encryption schemes were also designed to protect the privacy of each independent participant, whose benefits are: 1) sensitive information is kept from the server; and 2) accuracy is kept intact [104]. Specifically, homomorphic cryptosystems allow certain forms of computation performed on ciphertexts and produce encrypted results, which matches the results of operations conducted on the plaintext. Furthermore, homomorphic cryptosystem is mainly classified into two categories: partially homomorphic cryptosystem with either additive or multiplicative homomorphism, and fully homomorphic cryptosystem supports arbitrary computation on ciphertexts. Under the context of federated learning privacy protection, the stochastic gradient descent can be protected with additive homomorphic encryption, when the server is assumed to be collusion-resistant and honest-but-curious. That is, the honest-but-curious server is restricted to follow the designed protocol, but it may attempt to recover the content from the gradient information. Another hybrid scheme combining differential privacy technique and threshold homomorphic encryption is also designed, which can further resist collusion attacks between the colluding server and participants [105].

Note that in our proposed scheme, no matter which distributed model training method (FedAvg or cADMM or pxADMM) is involved, we utilize a homomorphic encryption algorithm for the mobile users to upload the local gradients or hyper-parameter towards the server. In order to illustrate the stochastic descent encryption and decryption process, we take the widely exploited additive homomorphic Paillier cryptosystem as an example [106], which consists of three algorithms: key generation, encryption and decryption.

Key generation: Given security parameter κ, two large prime numbers p, q are chosen, where $|p| = |q| = \kappa$. Then the Rivest-Shamir-Adleman (RSA) modulus $n = p \cdot q$ and the least common multiple $\lambda = \mathrm{lcm}(p-1, q-1)$ are computed. Define a function $L(u) = \frac{u-1}{n}$, after choosing a generator $g \in \mathbb{Z}_{n^2}^*$, $\mu = L(g^\lambda \mod n^2)^{-1} \mod n$ is further calculated. Then the public key is $pk = (n, g)$ and the private key is $sk = (\mu, \lambda)$.

Encryption: Given a message $m \in \mathbb{Z}_n$, choose a random number $r \in \mathbb{Z}_n^*$, and the ciphertext can be calculated as $c = E(m) = g^m \cdot r^n \mod n^2$, where set $\mathbb{Z}_n = \{0, 1, ..., n-1\}$, and set $\mathbb{Z}_n^*$ only has elements coprime to n.

Decryption: Given the ciphertext $c \in \mathbb{Z}_{n^2}^*$, the corresponding message can be recovered as $m = D(c) = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

For our federated learning scenario, the Paillier cryptosystem [106] can be exploited to protect and aggregate the stochastic descents. Interested readers can find more implementation details in [21].

Algorithm 1: FedLoc Framework under Cloud-Based Network Infrastructure

```
Input: (1) A massive number of collaborating mobile terminals with index $k = 1, 2, ..., K$; (2) Local data $\mathcal{D}_k = \{\boldsymbol{X}_k, \boldsymbol{y}_k\}$, where the inputs and outputs are positions/position related measures; (3) A learning model, for instance a DNN or a GP model.
Output: Optimal hyper-parameters $\boldsymbol{\theta}^*$ of the global learning model.
1  Initialization: Initial hyper-parameters of the selected learning model, $\boldsymbol{\theta}^0$; iteration index, $\eta = 0$.
2  for (outer iterations) $\eta = 0, 1, ...$ do
3      1. The core network sends a probing signal to all mobile terminals and identifies which ones are idle during this round. The idle terminals form a set, $\mathcal{K}_\eta$.
4      2. The core network sends a seed to the selected terminals for encoding the messages as well as the current hyper-parameter estimate, $\boldsymbol{\theta}^\eta$.
5      for (inner iterations) each idle mobile terminal $k \in \mathcal{K}_\eta$ in parallel do
6          1. Use the local data, $\mathcal{D}_k$, or a fraction of it to update the hyper-parameter $\boldsymbol{\theta}_k^{\eta+1}$, for instance, via FedAvg/FedProx for DNN or via cADMM/proximal ADMM for GP.
7          2. Encrypt the local update of the global model hyper-parameters as a message using for instance homomorphic encryption.
8          3. Send the encrypted message to the core network.
9      end
10     3. The core network receives all encrypted messages from the mobile terminals indexed in $\mathcal{K}_\eta$ and performs decryption.
11     4. The core network updates the global learning model hyper-parameters via consensus.
12     5. Finish this round and reset $\eta = \eta + 1$.
13     6. Repeat the above iterations (1)-(5) until certain stopping criteria are satisfied.
14 end
15 The approximated global hyper-parameters is $\boldsymbol{\theta}^* = \boldsymbol{\theta}^\eta$.
```

D. FedLoc: A New Umbrella of Old Modules

In the previous sections, we have introduced two important classes of learning models, namely the deep neural network models and Gaussian process models, and a few distributed hyper-parameter optimization schemes tailored to these two models, as well as the state-of-the-art privacy preservation methods for mobile data. These constitute the major ingredients of a novel cooperative, data-driven, learning model-based framework for localization and location data processing.

For clarity, we give a complete procedure of the FedLoc framework in Algorithm 1, which can be adopted for both the cooperative localization and the cooperative location data processing. Various live use cases in different application sectors already fall into or can be revised to suit our FedLoc framework. In Section VI, we will show a few representative use cases and survey some related works that can be made adapt to the FedLoc.

V. Network Infrastructures for FedLoc

As it is widely known, federated learning needs to communicate a big number of model parameters continuously over the air, especially when DNN is adopted as the learning model. In this section, we introduce two promising network infrastructures to meet the communication requirements of the proposed FedLoc framework. Specifically, a cloud-based wireless network infrastructure is introduced in Subsection V-A, while an emerging edge-based one is introduced in Subsection V-B. Some remarks are given in Subsection V-C. More fresh discussions on using parallel infrastructures to support scalable learning paradigms for data-driven wireless applications can be found in our recent work [111].

A. Cloud-based Infrastructure

For ease of understanding, a complete picture of the network infrastructure is depicted in Fig. 5 for learning model-based cooperative localization. The key elements of this network as well as their functionality are summarized as follows:

1) *Reference Network Node* is equipped with cache, storage, and communication entities. A reference network node communicates with the mobile terminals deployed in its communication range to exchange learning model related information. Both the position and the transmit power of a reference network node are assumed to be precisely known. Representative reference network nodes include 5G macro and micro base stations, WiFi access points, BLE beacons, etc. Especially the emerging 5G and WiFi-6 network are able to provide low-latency, high throughput wireless transmission to FedLoc, which requires to transmit a big amount of model parameters in every iteration. Table II gives some numbers. For better intuitions, two specific examples are given below. The 5G network with the highest throughput can support a 9-layer fully-connected DNN with the network layout “20000-30000-10000-100000-10000-10000-10000-1000-10” that has around 300 million weights. The 4G network, however, can only support an 8-layer fully-connected DNN with a much smaller network layout “5000-5000-10000-3000-9000-2000-200-10” with around 15 million weights.
2) *Mobile Terminal (MT)* is equipped with sensing, logging, computing, storage, and communication entities. Moreover, the MT has installed the designated mobile applications for carrying out the calibration work. The MT collects position related measurements, obtains a local update of the global learning model parameters, and uploads them to the core network. All the computations are conducted on-device using the local data only. Here, the mobile terminal refers to a smartphone specifically. It is noteworthy that modern smartphones are equipped with a basket of inertial sensors, including accelerometer, gyroscope, magnetometer, barometer, pedometer,

Wireless infrastructure	Max uplink data rate (Mbps)	Max downlink data rate (Mbps)	Number of DNN weights (Million)	Configuration
5G [107]	10,000	20,000	312.5	IMT-2020 peak rate
4G [108]	500	1000	15.625	IMT-advanced
WiFi-6(ax) [109]	2400	2400	75	160MHz 2*2MIMO 1024-QAM 802.11ax
WiFi-5(ac) [110]	1733	1733	54.16	160MHz 2*2MIMO 256-QAM 802.11ac

TABLE II
DOWNLINK AND UPLINK DATA RATE OF DIFFERENT WIRELESS INFRASTRUCTURES UNDER SPECIFIC CONFIGURATIONS AND THE NUMBER OF HYPER-PARAMETERS OF A SELECTED LEARNING MODEL (TAKING THE DNN WEIGHTS AS EXAMPLE) THAT CAN BE SUPPORTED. THE NUMBER OF THE DNN WEIGHTS (IN MILLION) SHOWN IN THE FOURTH COLUMN IS EQUAL TO THE UPLINK RATE (GIVEN IN THE SECOND COLUMN) DIVIDED BY 32 BITS PER DNN WEIGHT.

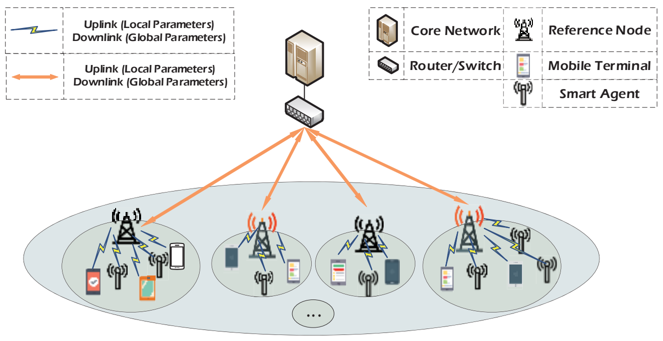

Uplink (Local Parameters)
Downlink (Global Parameters)
Uplink (Local Parameters)
Downlink (Global Parameters)
Core Network
Reference Node
Router/Switch
Mobile Terminal
Smart Agent
...

Fig. 5. Cloud-based network infrastructure for supporting the proposed FedLoc framework. For illustration purpose only, the whole deployment area is divided into many non-overlapping sub-areas, and for each sub-area there is a bunch of mobile terminals willing to collaborate.

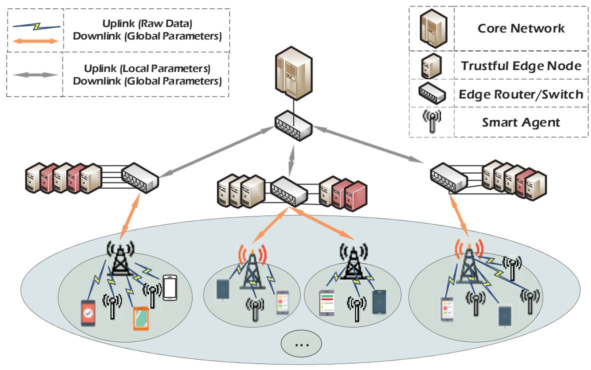

Uplink (Raw Data)
Downlink (Global Parameters)
Uplink (Local Parameters)
Downlink (Global Parameters)
Core Network
Trustful Edge Node
Edge Router/Switch
Smart Agent
...

Fig. 6. Edge-based network infrastructure for supporting the FedLoc framework.

barcode/QR code sensors, that can be exploited for localization or localization-related tasks. Apart from the rapid development of the hardware, a number of mobile machine learning platforms are under development, such as Tensorflow by Google, Core ML by Apple, Caffe2 by Facebook, Paddle Lite by Baidu, MNN by Alibaba, etc. Mobile users can easily deploy different deep learning models on their smartphones in the near future.

3) *Fixed Smart Agents* are equipped with sensing, logging, computing, storage and communication entities. Representative smart agents include IoT machines, wireless sensors, robots, smart traffic lights, unmanned aerial vehicles (UAVs), micro-base stations that are collecting location data continuously.
4) *Core Network* is equipped with high-speed computing, cache/storage and communication entity. The local updates from the mobile users are aggregated to the core network to compute a global parameter update. After the training phase is over, the approximated global learning model will be stored in the core network and used for predicting a new position in the online phase. Since the heavy computations have been offloaded to a number of mobile users, the core network can perform smarter coordination of different tasks and resources, so as to make the whole network agile and adaptive to the fast changing environments.

B. Edge-based Infrastructure

In the second infrastructure, the mobile users or smart agents can upload their local data to a trustful third-party edge node, where there is sufficient storage and computation power for handling learning tasks. For clarity, we show this network infrastructure in Fig.6. The edge node first pre-processes the received data and then offloads the model fitting task to a number of computing units. Each edge node is in charge of building a locally-global learning model and transmits the trained hyper-parameters to the core network for consensus and coordinated control. This infrastructure is more suitable for building a number of regional global models for location data processing. The third use case that we will show in the next section can potentially benefit a lot from this edge-based infrastructure.

It is noteworthy that our proposed FedLoc solution, as shown in Algorithm 1, can be easily extended to work with edge nodes. Specifically, we can simply let a batch of end devices upload their raw data to the nearest neighboring edge for training a regional local model; afterwards, all such edge nodes make consensus on the global model parameters by following similar steps given in Algorithm 1. In this way, the whole network can be made hierarchical, and the latency in each level can be minimized.

C. Concluding Remarks

First, we would like to stress again that the communication requirement is sufficient broadband for transmitting a large number of global model parameters between users and the cloud/edge, which is more challenging for the deep neural networks than GP models as mentioned in Subsection III-C. Under this communication constraint, people may be concerned about the necessity of using the two above-mentioned infrastructures and their corresponding pros and cons. Our understanding is that when the distributed device is equipped with a powerful computing unit and battery, then it is preferred to perform a sufficient local numerical search before communicating its local estimate of the model parameters. In this way, fewer optimization epochs may be required (or equivalently less communication overhead is required). This is suitable for the distributed FedLoc system to be deployed in certain areas with less developed wireless infrastructure, for instance, suburbs covered by 3G only. On the other hand, if the end devices, such as wireless sensors and smart meters, have rather low on-board computation capacity but are deployed in certain areas with a rather good network support, for instance, using 5G or private IoT network, then we could simply increase the rounds of communications to compensate for the quality of local updates obtained using economical numerical searches.

The overwhelming benefit of the cloud-based solution lies in the global model training accuracy. Principally, distributed systems can only approach its performance. But the major demerits of the cloud-based solution are also obvious, among others, that response latency is the most serious one for big data aggregation and dissemination even using 5G network. Therefore, a promising solution is to employ edge nodes for effective trade-off between computation and communication.

VI. Use Cases of FedLoc

This section aims to shed more light on the FedLoc framework with various live use cases. In particular, we showcase: (1) DNN-based static localization/fingerprinting; (2) DNN-based smartphone sensor calibration for accurate navigation with low-sampling-rate GPS; (3) GP-based state-space model for target tracking and navigation; and (4) GP-based wireless traffic prediction in 5G C-RAN. The first three use cases relate to localization, while the last one relates to location data processing and prediction. Most of the above use cases are summarized from our recent works. We also survey related works that can easily fit into the FedLoc framework.

A. DNN-Based Static Localization/Fingerprinting

There exist various statistical methods using wireless measurements, such as ToA, RSS, proximity [112], [113], for static target localization. These methods rely on empirical propagation models. In this subsection, we show a different static localization method using DNN, which can benefit from the federated learning framework. DNN-based localization is preferred for complex indoor wireless environments, for which sophisticated empirical models are either not available or incapable of capturing the underlying propagation mechanism.

Let us take a look at three representative indoor scenarios:

- **Indoor shopping mall**, where there are a bunch of WiFi/BLE access points and micro base stations for public data traffic. In addition, thanks to the rapid spread of 5G for IoT and machine-type communications (MTC), there are now a large number of machines/landmarks with QR codes in the shops. By scanning the QR codes, customers can easily get shopping mall information and promotional information. Some live examples are demonstrated in Fig. 7.

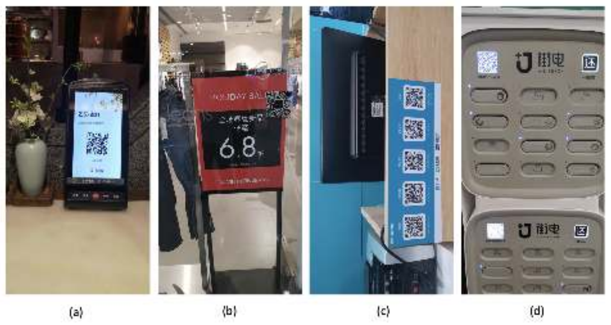


Fig. 7. All the QR labels were photoed in a modern shopping mall in Shenzhen, China. (a) QR codes for ordering foods for a specific dining table; (b) QR code for promotion information at a shop; (c) QR codes for various different services, including product recommendation, payment, etc at the cashier of a shop; (d) QR code for renting mobile power bank.

- **Indoor museum**, where there are a bunch of WiFi/BLE access points in the exhibition rooms and a considerable number of QR labels attached to the exhibits to serve as references. Similarly, by scanning the QR codes a visitor can easily get access to detailed interpretation of the exhibits on his/her mobile terminal.
- **Indoor office**, where there are a bunch of WiFi/BLE access points in the whole office area, and a large number of QR labels are placed on all valuable assets in the room.

The DNN-based static localization/fingerprinting needs to be trained with a big dataset $\mathcal{D}$, where the training input, $\boldsymbol{X}$, contains the radio features at different locations and the training output, $\boldsymbol{y}$ contains the corresponding locations. More concretely, let us assume that a training input comprises RSS measured with respect to P WiFi/BLE access points, $\boldsymbol{x}_i = [RSS_{i,1}, RSS_{i,2}, ..., RSS_{i,P}]$, and the output $\boldsymbol{y}$ is a position (2D or 3D) at which the radio feature is measured. More sophisticated measurements such as magnetic fields and channel state information (CSI) can be used instead of RSS or jointly used with RSS. Note that an output $\boldsymbol{y}_i$ is either measured precisely at the calibration points by paid workers or imprecisely (for instance, with the aid of the landmark points and manual click on the indoor map displayed on the mobile application) by voluntary users. In either case, we assume the output is subject to additional independent noise. A concrete example is illustrated in Fig. 8.

The regression problem can be formulated as

$$\boldsymbol{y}_i = f(\boldsymbol{x}_i; \boldsymbol{\theta}) + \boldsymbol{n}_i, \tag{28}$$

where $f(\boldsymbol{x}; \boldsymbol{\theta}) : \mathbb{R}^{dx} \to \mathbb{R}^{dy}$ represents a DNN with an input of $dx = P$ features and the neural network weights $\boldsymbol{\theta}$ to be tuned. The regression function $f(\boldsymbol{x}; \boldsymbol{\theta})$ is also known as RSS map or fingerprinting map in the literature.

In order to adopt the federated learning framework, we deploy a large number of mobile terminals, and each is responsible for a particular area, possibly overlapping with

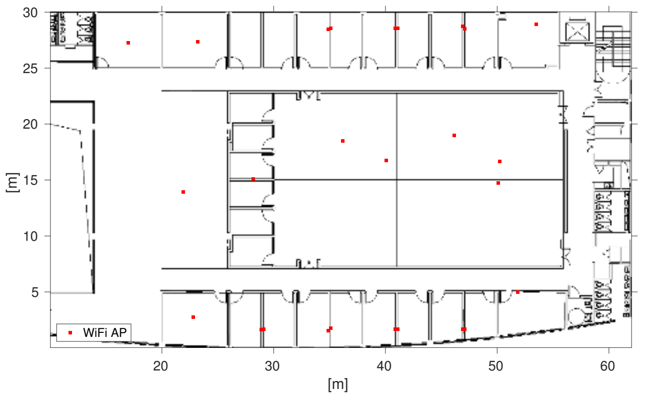


Fig. 8. A typical indoor office environment at the Chinese University of Hong Kong (Shenzhen), where two dozens of WiFi access points are deployed in the offices and laboratories. For this conceptual example, an input, $\boldsymbol{x}_i$, is a vector of $P = 26$ RSS values, and the corresponding output, $\boldsymbol{y}_i = [p_i^x, p_i^y]$ is a 2D position.

its neighboring areas. The k-th mobile terminal collects a dataset $\mathcal{D}_k = \{\boldsymbol{X}_k, \boldsymbol{y}_k\}$ and uses it to train a local update of the global parameters. Concretely, each mobile user solves

$$\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \sum_{\forall\{\boldsymbol{x}_i,\boldsymbol{y}_i\}\in\mathcal{D}_k} ||\boldsymbol{y}_i - f(\boldsymbol{x}_i;\boldsymbol{\theta})||_2^2. \tag{29}$$

All the mobile terminals cooperate to perform Algorithm 1. Since in this use case, the global objective is readily in the form of summation, therefore we can set the weights β_k to be the ratio $|\mathcal{D}_k|/\sum_{j\in\mathcal{K}_\eta}|D_j|$ in the η-th iteration and update $\boldsymbol{\theta}^{\eta+1} = \sum_{k\in\mathcal{K}_\eta}\beta_k\boldsymbol{\theta}_k^{\eta}$. When the messages are exchanged between the core network and mobile terminals, they are first encrypted by the mobile terminals and decrypted in the core network using homomorphic techniques. The workflow of the FedLoc for DNN-based static localization is shown in Fig. 9.

After the training procedure is terminated, the central node will obtain an approximated global estimate of the hyper-parameters, denoted by $\hat{\boldsymbol{\theta}}$. Given a new vector of RSS measurements, $\boldsymbol{x}_* = [RSS_{*,1}, RSS_{*,2}, ..., RSS_{*,P}]$, reported to the central node, the trained learning model will map it then to the desired position estimate through $\boldsymbol{p}_* = f(\boldsymbol{x}_*;\hat{\boldsymbol{\theta}})$.

Various works on using deep learning models and RSS measurements for indoor fingerprinting have been published in recent years, for instance [114], [115], [116], [117], [118] based on DNN, CNN, LSTM. Although these works are originally centralized algorithms, they can be implemented in a distributed manner under our FedLoc framework.

We have shown the general procedure of DNN-based static localization. One may be concerned about the latency of the proposed FedLoc framework using DNN-based models in practical use. We believe that the latency issue would not prohibit the wide use of sophisticated learning models in next-generation wireless networks. The reasons are as follows. First, federated global model training can be done in the offline phase with historical data, which does not raise any latency issue. Second, when the global model has been trained, performing position inference using the well-trained model is computationally cheap. For a neural network with a modest number of layers and neurons in each layer, the computational time for estimating a position is just slightly higher than the KNN method, which should satisfy the LTE latency requirement. Third, with the rapid development of AI chips [119], we believe that the aforementioned inference time can be significantly shortened in order to meet the low-latency requirement of the 5G network and beyond.

Before leaving this subsection, let us give a concrete example to shed more light on both the latency and the positioning accuracy of KNN and CNN model. In [114], a real office room is considered and a CNN model is applied to predict some unknown 2-dimensional positions $\boldsymbol{y}$. The selected CNN architecture consists of 3 convolution layers and each convolution layer has 5 kernels, followed by two fully connected layers with 10 neurons in each layer. For more details, interested readers can refer to the original paper [114]. In the test phase, a number of 260 data samples (of 24-dimensional) are fed into CNN. The results show that the inference time for a single data point takes 0.001s, with the mean error distance around 160cm. In contrast, for the KNN model with a regular fingerprint resolution, testing one data point only takes 0.0002s, but the mean error distance is up to 230cm, being larger than that obtained using CNN.

B. DNN-Based Vehicle Navigation with Low Sampling Rate GPS

For land vehicle navigation, combining the inertial measurement unit (IMU) and global positioning system (GPS) embedded in a smartphone is still the main-stream technical solution. The GPS can readily provide accurate vehicle positions when the majority of the satellite signals are in line-of-sight (LOS) propagation with relatively high RSS. On the other hand, the IMU assembles, primarily, a three-axis acceleration sensor and a three-axis gyroscope, to determine the position and velocity of a vehicle. The main functionality of the IMU is to provide vehicle positions with a much higher sampling rate ($>$ 50 Hz) between two consequent GPS position estimates (with 1 Hz by default). Unfortunately, when a vehicle enters into certain areas with severe signal blockage, the received GPS signal will be very weak or even undetectable, leading to significantly degraded position estimate. On the other hand, solely relying on low-end IMU measurements for high-accuracy navigation is impractical due to the sensor bias, scale-factor error, and other random errors that accumulate over time. How can we maintain a satisfactory positioning accuracy for the case that GPS signals are occasionally available for harsh wireless environments, such as in the city center or forest? We demand a smart solution with affordable computational complexity.

Towards this end, we introduce in this subsection a machine learning-based approach that can be implemented on commercial smartphones and is able to provide high navigation accuracy using low-end inertial sensors and low-sampling-rate GPS. Inertial sensors are used to continuously estimate the vehicle velocity and position at higher sampling rate, while low-sampling-rate GPS signals are used for IMU calibration occasionally (for example every 60 seconds). When the GPS signal is not available, we use pre-trained DNNs to calibrate the inertial sensor errors.

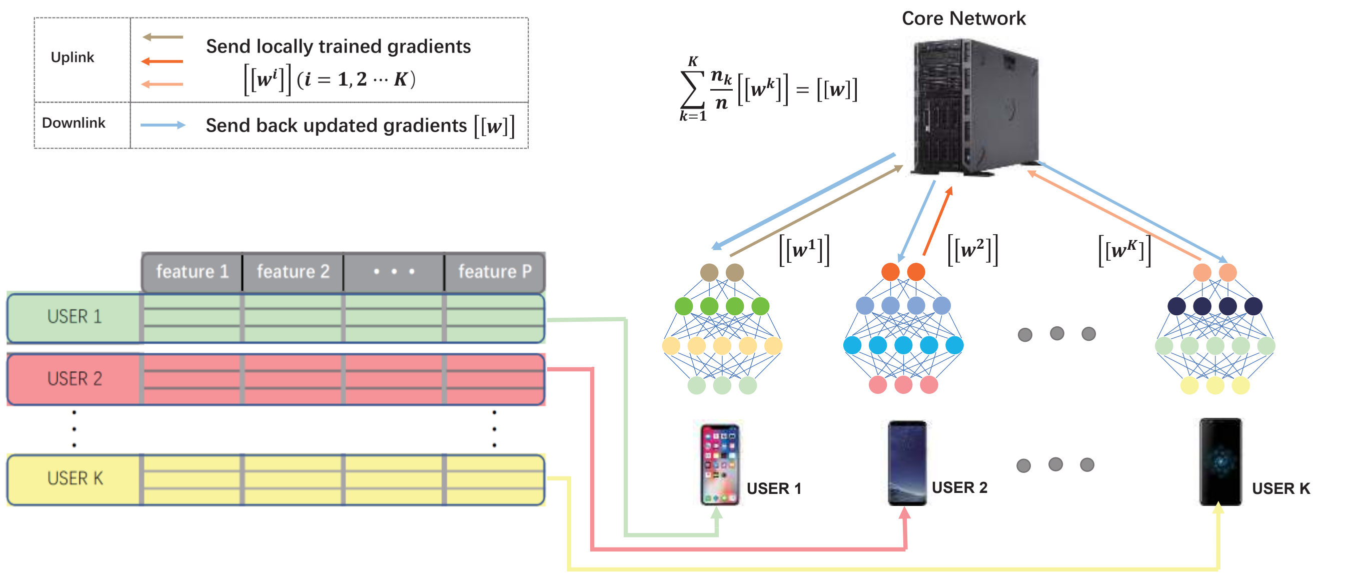

Uplink
Send locally trained gradients
$[[w^i]] \, (i = 1, 2 \cdots K)$
Downlink
Send back updated gradients $[[w]]$
Core Network
$\sum_{k=1}^{K} \frac{n_k}{n} [[w^k]] = [[w]]$
$[[w^1]]$
$[[w^2]]$
$[[w^K]]$
feature 1
feature 2
feature P
USER 1
USER 2
USER K
USER 1
USER 2
USER K

Fig. 9. Illustration of DNN-based static localization. Here, $[[W]]$ represents encrypted NN weight parameters using for instance Homomorphic Encryption (HE). The P features are RSS values collected from the WiFi access points in the deployed area.

To be concrete, we adopt two DNNs to estimate/predict the velocity $v_{t,NN1}$ and the yaw angle $y_{t,NN2}$ of the vehicle, respectively. In the model training phase, both DNNs take measurements from the smartphone inertial sensors as the input while the GPS velocity and yaw angle measurements are taken as the outputs/labels.

The first DNN takes the following inputs:

- The velocity $\widetilde{v}_t^n = ((v_t^{nx})^2 + (v_t^{ny})^2 + (v_t^{nz})^2)^{1/2}$ calculated from the inertial sensor data;
- The sequence of angular velocity $\{\omega_{t-l}^{bz}, ..., \omega_t^{bz}\}$ of the vehicle;
- The sequence of smoothed linear acceleration along the front direction of the vehicle, denoted as $\{a_{t-l}^{nx}, ..., a_t^{nx}\}$.

The DNN output is the velocity $v_{t,NN1}$ set to be the GPS velocity $v_{t,GPS}$ as the ground-truth in the training dataset.

Similarly, the second DNN takes the following inputs:

- The sequence of smoothed linear acceleration, denoted as $\{a_{by}, ..., a_{by}\}$;
- The sequence of angular velocity $\{\omega_{t-l}^{bz}, ..., \omega_t^{bz}\}$;
- The compensated yaw sequence $\{y_{t-l}, ..., y_t\}$.

The DNN output is the yaw angle $y_{t,NN2}$ set to the GPS yaw angle y_{tGPS} as the ground-truth in the training dataset.

Our recent work in [120] presented a centralized implementation, where interested readers can find more details about the measurements, configurations of the DNNs, as well as a diagram of the whole navigation system. In this paper, we are interested in designing a distributed counterpart. To this end, we let the two DNNs be trained individually by a batch of collaborating mobile users according to Algorithm 1 with the DNN weights optimized using either the FedAvg algorithm or the FedProx algorithm. The information exchange procedure remains the same as the first use case. In the online use phase, the two DNNs will calibrate the inertial sensor error aggregation when there is no GPS signal at hand. Some primary results for this use case will be shown in Section VII.

C. GP-Based State-Space Model (GPSSM) for Target Tracking

State-space models (SSM) are outstanding for modeling a time series $\boldsymbol{y}_{1:T} \triangleq \{\boldsymbol{y}_t\}_{t=1}^T$ with latent states $\boldsymbol{x}_{0:T} \triangleq \{\boldsymbol{x}_t\}_{t=0}^T$. An SSM comprises a transition function, $f(\boldsymbol{x}) : \mathbb{R}^{dx} \to \mathbb{R}^{dx}$ and a measurement function, $g(\boldsymbol{x}) : \mathbb{R}^{dx} \to \mathbb{R}^{dy}$. Concretely, an SSM is given by

$$\begin{aligned} \boldsymbol{x}_t &= f(\boldsymbol{x}_{t-1}) + \boldsymbol{e}_{t-1}, \\ \boldsymbol{y}_t &= g(\boldsymbol{x}_t) + \boldsymbol{n}_t, \end{aligned} \tag{30}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{dx}$ is the latent state, $\boldsymbol{y}_t \in \mathbb{R}^{dy}$ is the measurement, $\boldsymbol{e}_t$ is the process noise, and $\boldsymbol{n}_t$ is the measurement noise at time instance t, respectively. Traditional SSM restricts both the transition function $f(\boldsymbol{x})$ and the measurement function $g(\boldsymbol{x})$ to empirical, parametric functions [10], whose parameters can be learned through the expectation-maximization (EM) algorithm [121] or Markov chain Monte Carlo (MCMC) algorithm [122].

Since GP models provide outstanding performance in function approximation with a natural and inherent uncertainty region, they have been adopted to model complicated nonlinear functions in the SSMs, leading to the GPSSM [123]. Early variants of GPSSM were learned by finding the maximum *a posteriori* (MAP) estimates of the latent states, generating various successful positioning applications, among others the RSS-based WiFi localization [124], the human motion capture [125], and the IMU-based slotcar tracking [126], etc. The first fully probabilistic learning procedure of GPSSM was proposed in [127] using particle Markov Chain Monte Carlo (PMCMC). In order to reduce the heavy computational load of the sampling method used in [127], a number of different variational learning

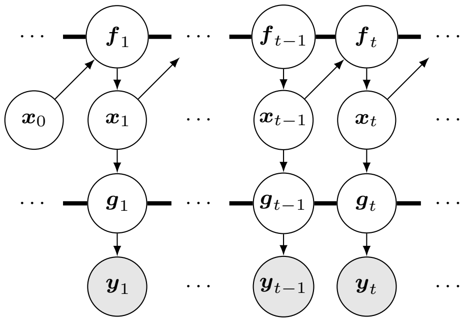


Fig. 10. Graphical representation of GPSSM. The shaded nodes represent the measurements, while the transparent nodes represent the latent variables. Variables belonging to the same GP are connected by a thick edge.

procedures were developed in [123], [128], [129], [130] upon the classical variational sparse GP framework [78].

A general GPSSM can be formulated as

$$\begin{aligned} f(\boldsymbol{x}) &\sim \mathcal{GP}(m_f(\boldsymbol{x}), k_f(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_f)), \\ g(\boldsymbol{x}) &\sim \mathcal{GP}(m_g(\boldsymbol{x}), k_g(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_g)), \\ \boldsymbol{x}_0 &\sim p(\boldsymbol{x}_0), \\ \boldsymbol{f}_t &= f(\boldsymbol{x}_{t-1}), \\ \boldsymbol{x}_t|\boldsymbol{f}_t &\sim \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{f}_t, \boldsymbol{Q}), \\ \boldsymbol{g}_t &= g(\boldsymbol{x}_t), \\ \boldsymbol{y}_t|\boldsymbol{g}_t &\sim \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{g}_t, \boldsymbol{R}), \end{aligned} \tag{31}$$

with the model hyper-parameters $\{\boldsymbol{\theta}_f, \boldsymbol{\theta}_g, \boldsymbol{Q}, \boldsymbol{R}\}$, where $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ are the kernel hyper-parameters of the GPs, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the covariance matrices of the process noise and the measurement noise, respectively. For clarity, Fig. 10 shows a graphical representation of GPSSM. In the following, we will first introduce the standard GPSSM, which requires a big set of calibrated data to train both the transition function f and the measurement function g. Then, we will briefly mention the advanced variational GPSSM proposed initially in [123].

We start with the transition function of the standard GPSSM. The GP regression model for the transition function, f, is $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{e}_t$, where the output $\boldsymbol{x}_{t+1} \in \mathbb{R}^{dx}$ is the state at time $t+1$, $\boldsymbol{x}_t \in \mathbb{R}^{dx}$ is the current state at time t, the unknown function $f(\boldsymbol{x}_t) : \mathbb{R}^{dx} \to \mathbb{R}^{dx}$ is essentially a multi-output GP [1], and $\boldsymbol{e}_t$ is a vector of noise terms. For simpler implementation, we could model each entry of the state, say the j-th, by an independent GP as $[\boldsymbol{x}_{t+1}]_j = f_j(\boldsymbol{x}_t) + e$, where $f_j(\boldsymbol{x}_t) : \mathbb{R}^{dx} \to \mathbb{R}$ is now a single-output GP. As discussed in Section III, we need to select a kernel function, $k_f(\boldsymbol{x}_t, \boldsymbol{x}_{t'}; \boldsymbol{\theta})$ to represent the correlation between the states at different time instances. When the input dimension is small/modest, using the ARD kernel is often a good choice. While for large input dimension, advanced kernels such as the arc-cosine kernel and the NTK should better be tried out.

The above GP models can be trained with a dataset of calibrated trajectories, $\mathcal{D}_j \triangleq \{\boldsymbol{X}, \tilde{\boldsymbol{x}}_j\}$, where $\tilde{\boldsymbol{x}}_j = [[\boldsymbol{x}_1]_j, [\boldsymbol{x}_2]_j, ..., [\boldsymbol{x}_T]_j]^T$ is a vector of outputs and $\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{T-1}]^T$ is a matrix of inputs. One could follow Eq.(18) to solve for the global ML hyper-parameter estimate. To implement the FedLoc framework, one could let K mobile users collaborate to approximate the global ML hyper-parameter estimate according to Eq.(19) with the local trajectories walked by each individual. The central node makes consensus on the local hyper-parameter estimates.

The GP regression model for the measurement function is $\boldsymbol{y}_t = g(\boldsymbol{x}_t) + \boldsymbol{n}_t$, where the input $\boldsymbol{x}_t \in \mathbb{R}^{dx}$ is the state at time t, the output $\boldsymbol{y}_t$ is a vector of wireless measurements, and the unknown function $g(\boldsymbol{x}_t) : \mathbb{R}^{dx} \to \mathbb{R}^{dy}$ is essentially another multi-output GP. Similar to the modeling of the transition function, we apply an independent GP for each single entry of the output. Training the measurement function is similar to that of the transition function, f, introduced above. Interested readers can find more details about using GPs to model f and g in [131], [132]. After the GPSSM is built, it can be combined with the celebrated particle filter or smoother [133] to reconstruct unknown trajectories. In [134], we proposed a practical real indoor navigation system prototype based on GPSSM and achieved improved navigation accuracy in various tests with smartphone sensory data. Moreover, we derived both the posterior- and parametric Cramer-Rao bounds for general nonlinear filtering problems based on GPSSM in [135].

One drawback of the above standard GPSSM lies in the need for a relatively large training dataset with calibrated latent states, which requires a large amount of labor force. To remedy this drawback, some recent works [123], [129] incorporated the variational inference technique [78] into the standard GPSSM to jointly estimate the GPSSM model hyper-parameters and the latent states on the fly. The variational GPSSM does not require a historical calibrated dataset, but as a tradeoff, it has to deal with a large-scale optimization problem. In order to make it adapt to the FedLoc framework, one may consider using the distributed variational inference techniques [136] with the GPSSM.

D. GP-Based Wireless Traffic Modeling and Prediction

In 5G, wireless traffic prediction is vital to resource allocation, load-aware management, and proactive control in C-RAN. In [11], we proposed a distributed GP-based wireless traffic modeling and prediction framework that exploits the advanced C-RAN specifying the edge-based network infrastructure given in Section V. In the deployment area, several hundreds of micro base stations with fixed geographical positions are installed to serve mobile users and record the downlink physical resource block (PRB) usage (a wireless traffic usage indicator) versus time. In this work, the base stations serving as smart agents are first clustered into groups according to their geographical locations, and for each group an aggregated PRB usage prediction model is to be built. To this end, all the micro base stations in one cluster send their observed time series of PRB usage to an edge node, in which the data are aggregated, pre-processed and uniformly allocated to a number of parallel computing units.

Specifically, a global GP regression model for the aggregated wireless traffic data of each cluster in the C-RAN is given as $y = f(t) + e$, where $y \in \mathbb{R}^1$ represents the PRB usage; e is a Gaussian distributed noise term with zero mean and variance σ_e^2; $f(t)$ is a temporal GP as introduced in Eq.(3) of Section III.

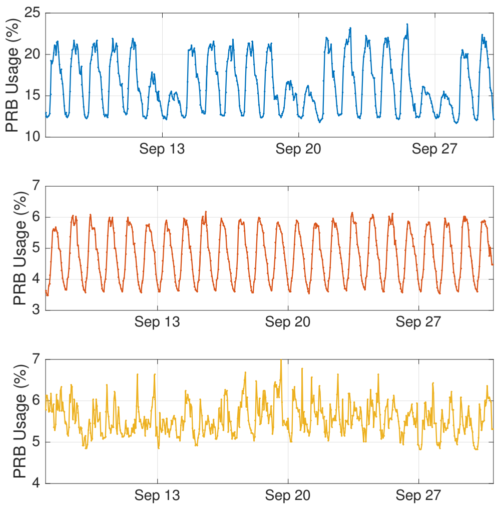


Fig. 11. The PRB usage curves of three base stations collected in three southern cities of China in 30 days. The data profile in the first panel reflects a typical office area, in which the traffic pattern shows a strong weekly periodic trend in accordance with weekdays and weekends. The data profile in the second panel reflects a typical residential area, in which the traffic pattern shows a strong daily trend with high demands in the daytime and low demands in the night. The data profile in the third panel reflects a typical rural area, in which the traffic pattern is more or less random.

In comparison with the "black-box" deep learning models for sequential data modeling such as the recurrent neural network (RNN) and long-short term memory (LSTM), GP model owns better interpretability as prior information about the wireless traffic pattern can be encoded more easily into the kernel function design. As shown in Fig. 11, the wireless traffic in our real datasets demonstrates the following general patterns: (1) *weekly periodic pattern*, namely the variation in accordance with weekdays and weekends; (2) *daily periodic pattern*, namely the variation in accordance with weekdays and weekends; and (3) *deviations*, namely the small scale variation in addition to the above periodic trends. The first two patterns can be well captured by the periodic or the locally periodic kernel, while the third pattern can be well captured by the SE kernel or the Matern kernel.

Our distributed GP for wireless traffic modeling and prediction falls in the FedLoc framework. Both the training and inference stages are performed in the edge nodes. Detailed workflow of model training is as follows. First, each base station in a specific cluster uploads its measured time series to the edge node. The aggregated data is then divided into K portions by the edge node, and each portion is allocated to a local computing unit for distributed model training based on the cADMM introduced in Section IV. The training framework achieves excellent tradeoff between the communication overhead and modeling accuracy, as explained in Section III. For each local computing unit, the required computational complexity can be reduced from $\mathcal{O}(n^3)$ of the centralized, standard GP to $\mathcal{O}(\frac{n^3}{K^3})$, where n is the number of the data points and K the number of parallel computing units.

In the online phase, one could use the generalized PoE [77] to fuse the local predictions from all parallel computing units to approximate the global prediction. The generalized PoE model needs to introduce a set of fusion weight parameters, β_i, $i = 1, 2, ..., K$, to take into account the importance of the local predictions. The resulting PoE predictive distribution is

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) \approx \prod_{i=1}^{K} p_i^{\beta_i}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(i)}). \tag{32}$$

The choice of β_i, $i = 1, 2, ..., K$, is vital to the prediction. In [11], we proposed to optimize the fusion weights according to the cross-validation criterion. The corresponding weight optimization problem can be solved efficiently with convergence guarantee. More details about the optimization process can be found in [11].

In the above work, we considered a temporal GP for regression. Therein, each cluster of base stations is assumed to be independent other clusters. For enhanced prediction performance, we could use spatio-temporal GP that takes into account the correlations between different clusters. A straightforward way for building a spatio-temporal GP model is to introduce an extra kernel to account for the spatial correlations between different clusters and combine this spatial kernel with the aforementioned temporal kernel either through addition [137] or Kronecker product [138].

The recently proposed graph GP provides another way for learning from high-dimensional data points living on non-Euclidean domains, see for instance [139], [140], [141]. As such, graph GP allows for better non-local generalization thus can be used to model sophisticated correlation patterns across time and space. In the illustrating example in Fig. 12, a graph GP can be designed to capture three types of correlations, including: (1) temporal correlation as discussed above; and (2) spatial-temporal correlation, where closer geographical distance indicates higher correlation in the temporal observations, and (3) the event correlation, where an event nearby also indicates a higher probability of an abrupt traffic change. It is noteworthy that graph GP is still under development where many directions remain to be explored, e.g., kernel design, stability issue, and distributed processing among others.

E. Other Potential Use Cases

Due to space limitations, we are unable to give a full list of all FedLoc related use cases with details. However, we want to briefly demonstrate the following three use cases due to their ever-increasing popularity.

(1) Radio feature map construction. The proposed FedLoc framework can be used by a number of collaborating mobile users to build accurate radio feature maps, such as RSS map and magnetic field map, for indoor venues. In [132], we proposed a distributed, recursive GP framework for building indoor RSS maps. Therein, a batch of mobile users was employed to collect RSS measurements from a dozen of WiFi access points at Ericsson research, Linkoping, Sweden. In the training phase, each mobile user trains a local GP empowered RSS map individually, while in the inference phase a global prediction is obtained by fusing all the local GP models via the classical Bayesian committee machine. A follow-up work was then proposed in [142]. These works can be revised to fit a global GP

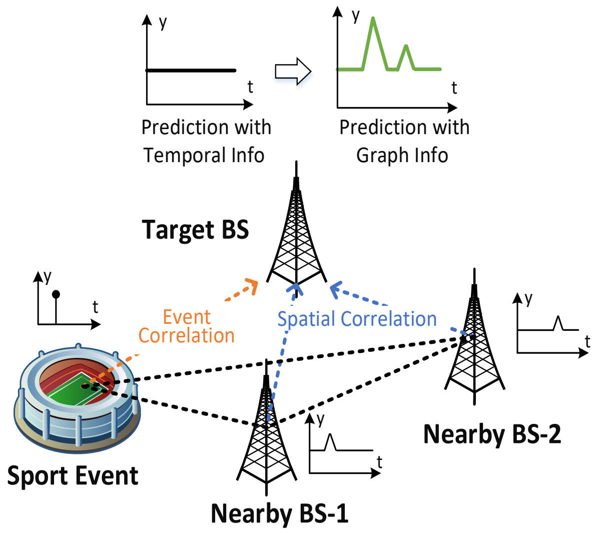


Fig. 12. Conceptual illustration of graph GP for spatio-temporal data modeling. Both the spatial correlation and event correlation information are helpful to improve the prediction performance.

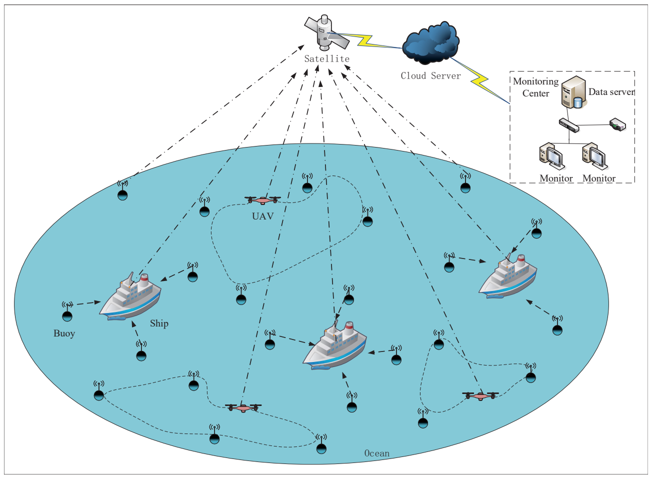


Fig. 13. A conceptual picture of Ocean-of-Things. Here, buoys can be seen as smart agents, ships and UAVs as edge nodes, and satellite as central node with cloud facility.

model in the training phase using the ADMM-based GP hyper-parameter optimization algorithm introduced in Section IV.

(2) Simultaneous localization and mapping (SLAM) for three-dimensional (3D) indoor scenario construction. The proposed FedLoc framework can also be used for a number of collaborating robots or low-flying unmanned aerial vehicles (UAVs) equipped with cameras and LIDAR to reconstruct a 3D indoor scenario. A generic SLAM model [133] is given as follows:

$$\begin{aligned} \boldsymbol{x}_t &= f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) + \boldsymbol{e}_{t-1}, \\ \boldsymbol{m}_t &= \boldsymbol{m}_{t-1}, \\ \boldsymbol{y}_t &= g(\boldsymbol{x}_t, \boldsymbol{m}_t, \boldsymbol{u}_t) + \boldsymbol{n}_t, \end{aligned} \tag{33}$$

where the dynamic motion model takes an additional inertial input $\boldsymbol{u}_t$ of the sensory data from odometer, accelerometer, gyroscope, and there is an additional map memory state, $\boldsymbol{m}_t$, in which the positions of the landmarks are updated and stored. We could potentially modify the GPSSM framework for the federated SLAM. Different from the use cases given in Section VI, federated SLAM imposes more stringent requirements on both the computational power of the mobile devices and the data throughput of the network, when dealing with 3D environment reconstruction. The commercial 5G network and futuristic wide-band generations (B5G and 6G) could make the federated SLAM possible. Some recent attempt in this regard can be found in [143].

(3) Ocean-of-Things (OoT) [144]. So far we have solely considered ground applications. In addition, there will be a plethora of emerging OoT applications that can benefit from our FedLoc framework. We show a conceptual picture of OoT in Fig. 13, where the whole network comprises a large number of spatially distributed buoys, some moving ships and UAVs, and satellites. The buoys are analogous to micro base stations on the ground, serving as smart agents, and they can perform data collection and monitor local environment. New fashioned buoys will be equipped with different sensors, ranging devices, GPS, and low-profile AI chips. They can be used to measure the ocean surface temperature, sea state, sound speed, etc, and track multi-target trajectories. The measured local data can be uploaded either to a moving ship or a moving UAV, which can be regarded as edge node. In addition to information transmission, the UAVs can also be used to charge the buoys if they are wireless powered [145], [146]. Each edge node maintains a local update of the learning model for spatio-temporal data processing and transmits the hyper-parameter estimate to the satellite cloud for consensus. In contrast to the ground IoT applications, the buoys may have insufficient on-board processing capability and relatively short communication range compared with a micro base station. However, the communication channels on the sea are mostly in line-of-sight. Since the buoys may be owned by different operators, privacy-preservation can not be ignored either.

VII. Results

In this section, we show the effectiveness of the FedLoc framework with two examples evaluated using real datasets. In the first example, we adopt GP as the learning model and mainly focus on the effectiveness of the distributed training of a small batch of model hyper-parameters. In the second example, we adopt DNN as the learning model and focus on practical implementation aspects.

A. *GPSSM for Indoor Target Tracking*

In this section, we will demonstrate the first example of applying the FedLoc framework for target tracking. The experimental setup aims for a quick and practical deployment of the framework, thus may not be theoretically optimal. Our focus is on both the training and prediction performance of the global, centralized model versus its distributed approximation under the FedLoc framework.

Due to space limitations, we will only show some results for the transition function in GPSSM. The model is $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{e}_t$, where the vector $\boldsymbol{x}_t = [x_t, y_t]^T$ contains the 2-D position of a pedestrian at time instance t. We apply individual GPs for each dimension, namely, we let

$$x_{t+1} = f_x(\boldsymbol{x}_t) + \boldsymbol{e}_{x,t}, \tag{34a}$$

$$y_{t+1} = f_y(\boldsymbol{x}_t) + \boldsymbol{e}_{y,t}, \tag{34b}$$

where both $f_x(\boldsymbol{x}_t)$ and $f_y(\boldsymbol{x}_t)$ are modeled by GP; for instance, we let

$$f_x(\boldsymbol{x}_t) \sim GP(m_x(\boldsymbol{x}_t), k_x(\boldsymbol{x}_t, \boldsymbol{x}_{t'})). \tag{35}$$

For clear exposition, we let the mean function $m_x(\boldsymbol{x}_t)$ be zero and the kernel function $k_x(\boldsymbol{x}_t, \boldsymbol{x}_{t'})$ be the ARD kernel, i.e.,

$$k_x(\boldsymbol{x}_t, \boldsymbol{x}_{t'}) = \sigma_{s,x}^2 \exp\left[-\frac{(x_t - x_{t'})^2}{l_{xx}} - \frac{(y_t - y_{t'})^2}{l_{xy}} \right], \tag{36}$$

where the kernel hyper-parameters are $[\sigma_{s,x}^2, l_{xx}, l_{xy}]^T$. For the y-dimension, we adopt a similar ARD kernel, $k_y(\boldsymbol{x}_t, \boldsymbol{x}_{t'})$, but with a different set of kernel hyper-parameters $[\sigma_{s,y}^2, l_{yx}, l_{yy}]^T$.

The above GP models can be trained globally with a training dataset $\mathcal{D}$ via the global, centralized maximum-likelihood estimation shown in Eq.(18). We know from Section III that the computational complexity scales as $\mathcal{O}(n^3)$ for centralized model training. Using the FedLoc framework is beneficial. On the one hand, mobile users can collect their own local training data without worrying about the data leakage issue, which may effectively encourage more people to collaborate. By adopting the cADMM or the pxADMM introduced in Section IV to approximate the global model hyper-parameters in a distributed manner, the overall computational complexity can be reduced to $\mathcal{O}(n^3/K^3)$, where K is the number of the collaborating mobile users. This work can be seen as a collaborative, data-driven method for learning the human walking trajectory, which is valuable for us to understand the behavior of pedestrians and predict their future positions.

To evaluate the performance of the FedLoc, we collected a dataset in a live indoor office environment, as was shown in Fig. 8. This dataset contains more than 50 trajectories with around 25,000 samples. In the training phase, three mobile users each collected 15 trajectories. Each mobile user obtained an approximation of the global GP model shown in Eq.(18) using its local 15 trajectories. In the test phase, we use the model hyper-parameters trained from the FedLoc to perform posterior prediction of the next state given a novel current state.

We compare two distributed GP hyper-parameter optimization schemes: (1) pxADMM-GP with the regularization parameters $\rho_i = 500$ and $L_i = 5000$, $\forall i$; and (2) cADMM-GP with $\rho_i = 500$, for $i = 1, 2, 3$. We set the values for ρ_i and L_i empirically. We consider convergence when the difference in all optimization variables between two consequent iterations is within 10^{-3}. The computer program was implemented using MATLAB and executed on an ordinary computer with 4 cores.

We show the model training results for both dimensions (x and y) in Fig. 14 and Fig. 15. The distributed schemes converge to different model hyper-parameter estimates compared with the ones trained centrally for the global model. One reason

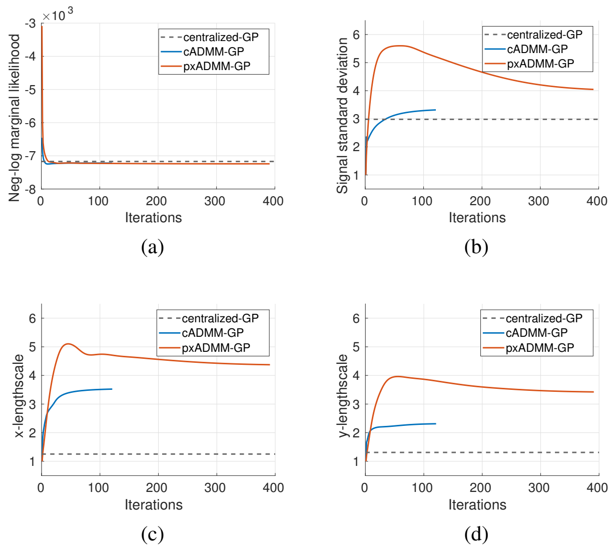


Fig. 14. For GP modeling along the x-dimension, we show the negative log-marginal likelihood functions (centralized formulation refer to Eq.(18) and distributed formulation refer to Eq.(19)) in sub-figure (a); and the ARD kernel hyper-parameter estimates as a function of training iterations for the 3 input variables using pxADMM-GP and cADMM-GP in sub-figures (b-d) for model variance, length-scale in x, and length-scale in y, respectively.

TABLE III
COMPARISONS OF TWO DISTRIBUTED GP MODEL TRAINING SCHEMES.

	pxADMM-GP	cADMM-GP
RMSE	0.1368m	0.1353m
CT	714s	10838s

is that the distributed scheme uses a different cost function as shown in Eq.(19), which corresponds to approximating the kernel matrix $K(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta})$ to a block diagonal matrix. Despite the difference in the hyper-parameter estimates, the corresponding negative log-marginal likelihood as well as the overall prediction root-mean-squared-error (RMSE in meters) are fairly close. From the computational time (CT) shown in Table III, we observed that the pxADMM-GP scheme consumed the least computational time. On one hand, the pxADMM-GP scheme circumvents frequent gradient synchronizations and used less iterations toward convergence than the cADMM-GP scheme. On the other hand, the closed-form proximal update w.r.t. the local hyper-parameters only requires to compute the expensive matrix inversion once.

We have shown above the distributed training of the GPSSM model. Next, we would like to show the positioning accuracy of the GPSSM-based tracking method compared with the empirical WiFi-based localization method and the pedestrian dead reckoning (PDR) method. To this end, we conduct experimental comparisons in a 1600 m^2 office environment, as depicted in Fig. 8. We aim to recover a "U"-shape walking trajectory of a pedestrian holding a smartphone. The geographical layout of the 26 WiFi APs in the deployment area is shown in Fig. 8. We developed a mobile application on a HUAWEI smartphone running Android 7.0 operating system to collect WiFi RSS measurements as well as IMU readings. We adopt a threshold value for RSS collection according to [147]. A sampling rate

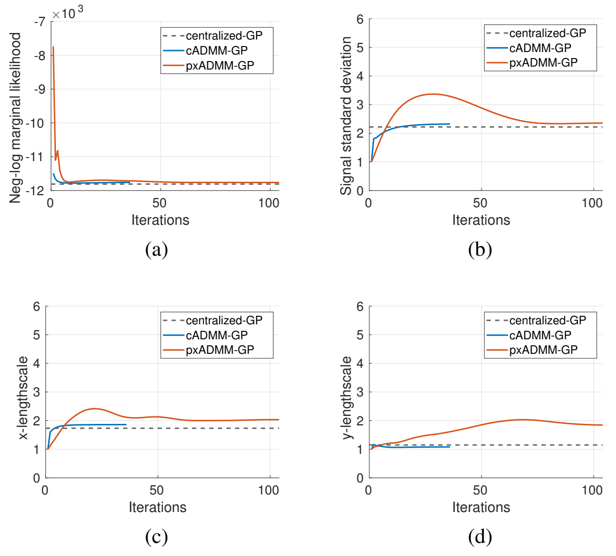


Fig. 15. Convergence results for the GP modeling along the y-dimension.

TABLE IV
TRACKING ACCURACY IN DIFFERENT METRICS.

	MAE	RMSE	STD	Largest Err.
WiFi Localization Only	3.55 m	5.04 m	3.58 m	15.94 m
PDR Only	5.34 m	5.42 m	0.96 m	7.52 m
GPSSM (1 Traj)	2.29 m	2.69 m	1.43 m	6.63 m
GPSSM (5 Trajs)	2.11 m	2.45 m	1.26 m	5.80 m

of 100 Hz is specified for the IMUs. The collected data are transmitted via wireless links to a computing server for further processing.

For practical usage, we reduce the optimization complexity in GPSSM by handling the measurement function and transition function in a sequential manner, thereby alleviating non-identifiability issues. Interested readers can find our latest work in [134] for more details. Therefore, to learn the measurement function g with a GP model, a total number of $N = 2059$ ground-truth positions and their corresponding WiFi localization results were recorded across the whole area. To learn the transition function f with a GP model, we recorded the WiFi localization results as well as the PDR control inputs while walking in the area. We repetitively record up to 5 trajectories along the same predefined path, and recover the first state trajectory that is tracked for 147 steps. For both f and g, we chose the standard squared exponential kernel with automatic relevance determination [1] in the GP models.

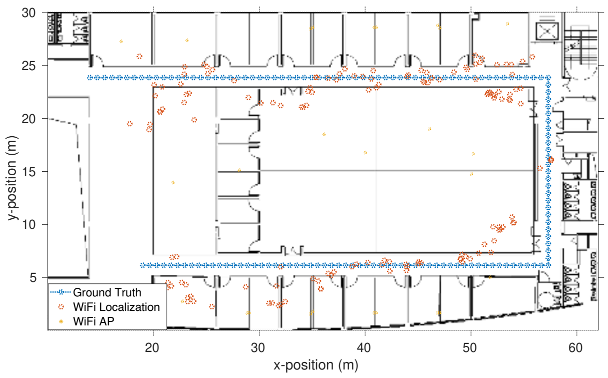
In Fig. 16, we show the state trajectories recovered by solely using the WiFi localization, PDR, and the fusion results of the GPSSM learned with data up to 5 trajectories. The corresponding tracking accuracy in terms of mean-absolute-error (MAE), root-mean-square-error (RMSE), standard deviation (STD), and the largest error are reported in Table IV. The MAE demonstrates that the GPSSM method reduces the mean tracking error by up to 40.6% and 60.5%, respectively, when compared to WiFi localization and PDR. Fig. 16a and 16b illustrate the performance two empirical models. Specifically, the WiFi localization gives unsatisfactory position estimates in the trajectory segment around the two sharp turns. This is due to the lack of APs deployed in that area and the unreliable RSS values received from far-away APs. PDR recovers a drifted state trajectory compared to the ground-truth. Since PDR only provides relative information, we use the WiFi localization result as the initial position for PDR. Clearly, both the WiFi localization and PDR are unsatisfactory. However, fusing WiFi localization and PDR in GPSSM takes the advantages of the individual techniques, hence achieves higher tracking accuracy. Fig. 16c and Fig. 16d show that the GPSSM keeps improving the estimation quality when learning over more training data. The outstanding modeling capacity of the GPSSM is exploited by feeding the model with large datasets, which contains comprehensive information about the indoor environment as well as pedestrian's motion patterns.

B. *Outdoor Vehicle Navigation with Low-Sampling-Rate GPS*

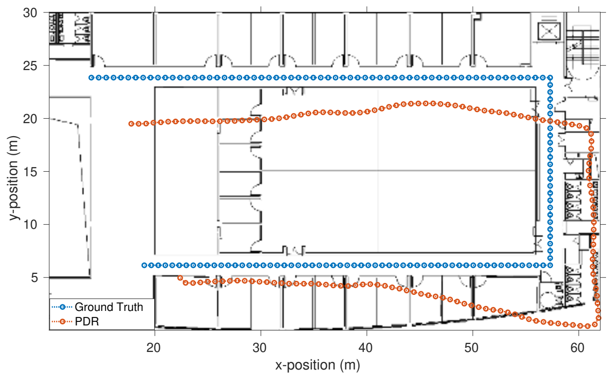
In this section, we will demonstrate the application of FedLoc with DNN models for smart vehicle navigation using low-sampling-rate GPS signals, which was introduced as a representative use case in Section VI.

We start by introducing the implementation setups of our new proposed federated learning empowered navigation system prototype. First, real datasets (for both training and test) were collected by three collaborating users with their own private car driving on the campus of The Chinese University of Hong Kong (Shenzhen), see Fig. 17. During the data collection process, each car was equipped with a smartphone (Xiaomi), facing upwards and heading to the moving direction of the car. The sensor data were uploaded to the server through WiFi on the fly. These three collaborating users traveled around the campus and collected various trajectories of smartphone sensory data that contain real-time motion information of their vehicles. The duration of each trajectory ranges from a few minutes to dozens of minutes.

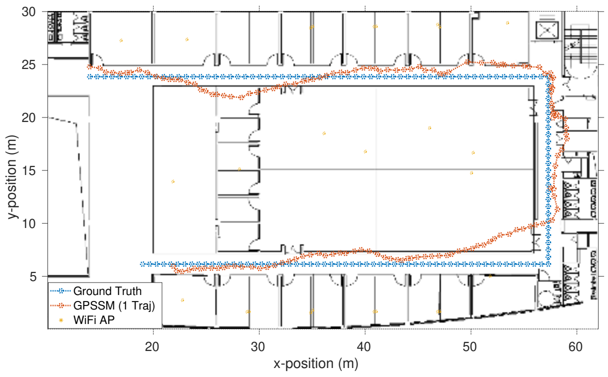
After collecting all training datasets, we adopted the FedLoc framework to train the two DNNs as was introduced in Section VI for calibrating the sensor data, one for the velocity and the other one for the yaw angle, so that accurate navigation can be obtained even with low-sampling-rate GPS signals. Two DNNs with five hidden layers (3000-3000-2000-1000-500) are selected as the global model in our prototype, which can be replaced with more sophisticated models, such as the LSTM, for high-dimensional time series. The input is the sensor data measured in a specific time window with dimension 600 for the first DNN or with dimension 401 for the second DNN, while the output is a scalar. In the training phase, the global model is updated by the three collaborating users according to Algorithm 1. Specifically, we tried two different model training algorithms, namely the FedAvg algorithm and the FedProx algorithm introduced in Section IV. We set the learning rate to 10^{-4} for both the FedAvg and FedProx algorithms. For the FedProx algorithm, the additional regularization parameter is set to 10^{4}. In the following, we consider two different experimental setups to mimic near i.i.d. and balanced data across the users



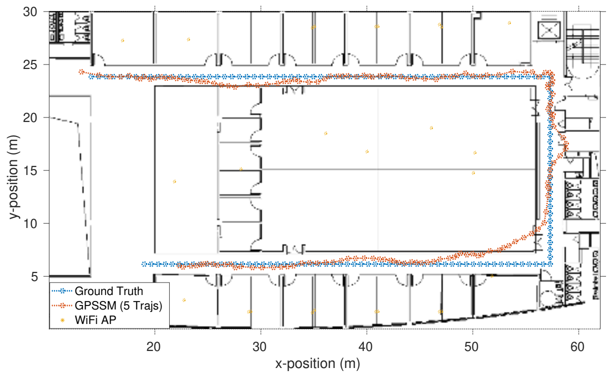
(a) WiFi Localization Only (Static Estimates)



(b) PDR Only



(c) GPSSM with 1 Trajectory Data



(d) GPSSM with 5 Trajectory Data

Fig. 16. Layout of the WiFi APs (yellow stars), the ground-truth trajectory (blue dots), and the recovered trajectories (orange dots) by using different models.

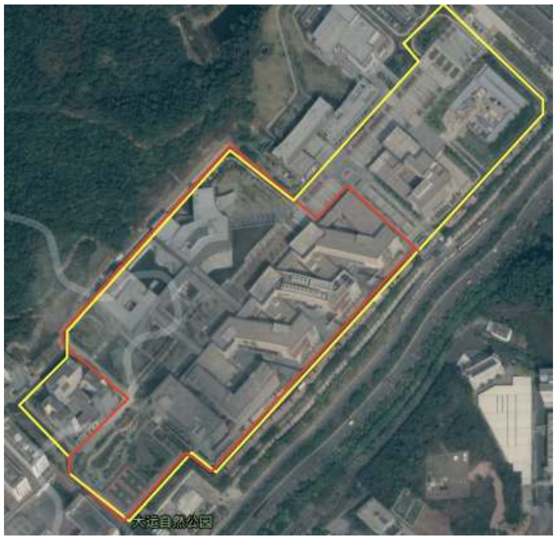


Fig. 17. The satellite map of the CUHK(SZ), where we collected the outdoor vehicle navigation data along two different routes.

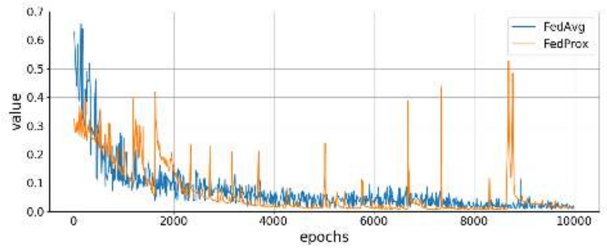
as well as non-i.i.d. and unbalanced data across the users, to test the FedLoc framework. We elaborate on the two different setups in Table V.

We show the training performance of both the FedAvg and FedProx algorithms in Fig. 18. Both algorithms can achieve a low training loss after a certain number of epochs. In our experiments, the FedProx algorithm unfortunately did not demonstrate smoother and more stable convergence profile than that of the FedAvg algorithm. The reason may lie in the improper setting of the regularization parameter of the FedProx algorithm, which is supposed to help achieve good trade off between the training loss and the discrepancy between the global model and local ones.

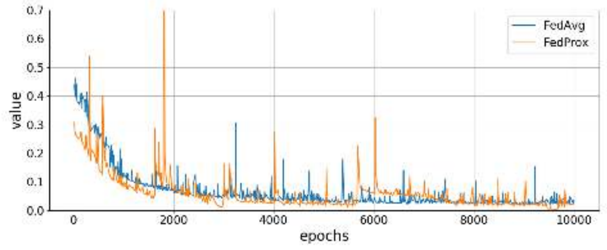
TABLE V
TWO DIFFERENT EXPERIMENTAL SETUPS.

		user 1	**user 2**	**user 3**
i.i.d. & balanced data	**route 1**	4	4	4
	route 2	0	0	0
non-i.i.d. & imbalanced data	**route 1**	0	2	6
	route 2	2	0	1

Lastly, we test the trained global learning model with two new trajectories of route 1. The GPS reference signals are only available every 60 seconds, being much less frequent than the default setup (1 sample per second). During the time where there is no GPS signal available, the trained global learning models are used to calibrate the observed sensor data. For the i.i.d. and balanced data setup mentioned in Table V, we show the test performance in Fig. 19. For this case, the FedAvg algorithm is modestly superior to the FedProx algorithms in the test phase. The navigation RMSE of the FedAvg is around 9 meters, while around 12 meters for the FedProx algorithm on average. Fine-tuning the learning rate of

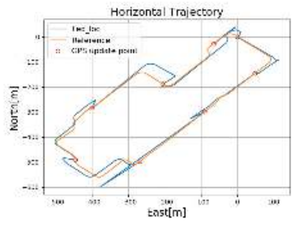


(a)

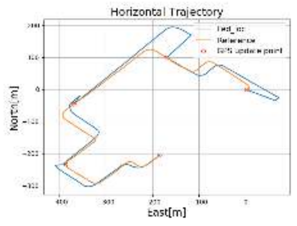


(b)

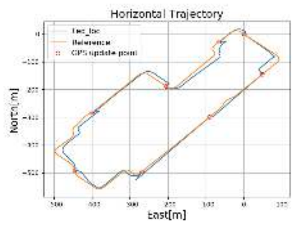
Fig. 18. Training loss versus optimization epochs for two different optimization algorithms. (a) Setup 1: near i.i.d. and balanced data; and (b) Setup 2: non-i.i.d. and imbalanced data.



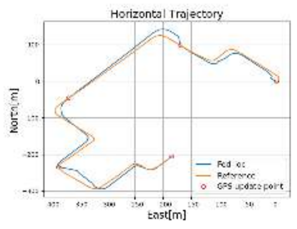
(a)



(b)



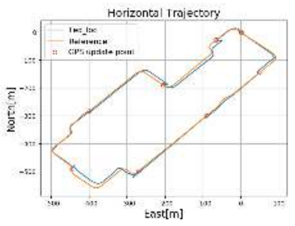
(c)



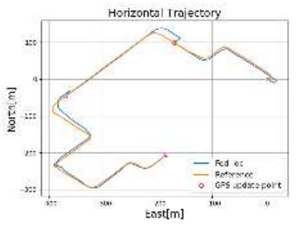
(d)

Fig. 20. The test performance on two test trajectories provided by the two algorithms for non-i.i.d. and imbalanced data setup. Subfigures (a) and (b) are drawn for the FedAvg algorithm; Subfigures (c) and (d) are drawn for the FedProx algorithm.

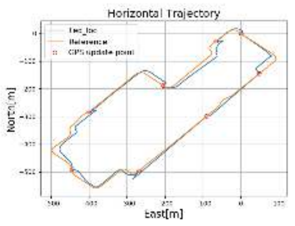
the FedProx algorithm may further improve its generalization performance. For the non-i.i.d and imbalanced data setup shown in Table V, it is obvious that the FedAvg algorithm failed with a significantly degraded navigation RMSE equal to 34 meters, while the FedProx algorithm worked well with a navigation RMSE around 17 meters. We show the test performance in Fig. 20. For both cases, using either the FedAvg algorithm or the FedProx algorithm leads to largely improved navigation RMSE compared with 90 meters when solely using the IMU for navigation.



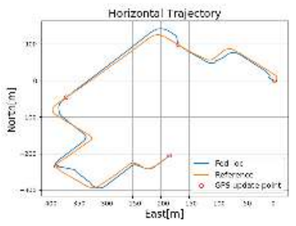
(a)



(b)



(c)



(d)

Fig. 19. The test performance on two test trajectories provided by the two algorithms for i.i.d. and balanced data setup. Subfigures (a) and (b) are drawn for the FedAvg algorithm; Subfigures (c) and (d) are drawn for the FedProx algorithm.

VIII. Future Directions and Challenges

Potential challenges to the federated localization are the following:

- An essential ingredient of the federated wireless localization framework is the mobile terminals. To ensure that the whole framework works smoothly, the mobile terminals should be able to process a modest amount of data and perform analysis with TensorFlow, PyTorch, etc. This requires further development of powerful but compressed deep learning models, mobile AI chips, etc. Advanced WiFi and 5G technologies can fulfill the communication requirements between the mobile terminals and the central node. However, communication efficiency is a critical issue that requires more attention. In addition, an agreement on the standard protocolclos for synchronizing the mobile terminals is to be made. Interested readers may refer to a recent work [148] on how to design a scalable production system for federated learning.
- In Section III, we mentioned that using DNN as the learning model will cause a lot of model parameters or gradients to be communicated over the air. A more straightforward and practical way to reduce the communication burden is to quantize the DNN weights from 64 bits precision to 8 bits precision or even lower. In the context of distributed optimization, a signSGD method was proposed in [149] that quantizes every gradient update to its binary sign thus reducing the communication load by a factor of 32. However, better understandings on the converge properties of such methods under practical setup, such as non-i.i.d. data distribution and imbalanced data size across mobile users, need to be built.
- The federated learning framework requires mobile users to cooperate. However, there might be the case that some voluntary mobile users are malicious or careless with

their shared messages. A promising way to solve such issues from the algorithmic perspective is to use robust distributed optimization [150], [151], robust estimation [152], and robust fusion [133] techniques for remedy.

- We have so far implicitly assumed that all the mobile users have sufficient number of local data for updating the global model hyper-parameters. This may not be true for voluntary users with very limited amount of local data. One effective way to alleviate this "small data" difficulty from algorithmic perspective is to harness the full basket of known canonical parametric models to generate some virtual data and mix them with the small batch of real data before training the model. In this way, we are able to transfer the prior knowledge of the canonical models to our desired data-driven, learning-based model [153].
- We have talked exclusively about wireless localization. Actually, visual-based localization and target tracking have also attracted a lot of attention these days. The combination of wireless measurements and visual measurements can effectively improve both the localization accuracy and the robustness. For instance, in [154] wireless positioning was adopted in visual trackers to alleviate visual tracking pains, such as long-term tracking, feature model drifting, and recovery. Their combination is a key enabler for autonomous driving and other robotic applications. However, the inhomogeneous data structure is a big challenge to federated learning.
- Another interesting direction is to exploit semantic information for indoor wireless localization [155], [156]. For example, in [155], authors improved the calibration of indoor wireless propagation models with the aid of the semantic target-environment relation information. This combination, in an unsupervised and automatic way, is considered to be a promising solution when dealing with a complex indoor environment.
- One could utilize the social relationship of mobile users to invite more participants to join the learning process and stimulate the activeness of current participants. To this end, graph learning models, for instance graph neural network [61] and graph GP [140], can be adopted for efficient learning from graph-like structured datasets.

IX. Conclusion

In this overview paper, we reviewed all required building blocks of a fundamentally new cooperative localization and location data processing framework, called FedLoc. Being different from most of the overview papers, we put more effort on real use cases of the FedLoc framework as well as their practical implementations. We strongly believe that the FedLoc framework is promising for the following good reasons. First, high-precision wireless localization is desperately demanded, which can be achieved by combining empirical models with data driven models. Second, calibrating a localization algorithm often consumes a lot of time and workforce, and collaboration among mobile users can largely facilitate the calibration effort. Third, smartphones are becoming a powerful platform for heavy computations. Fourth, we have seen rapid development in large-scale non-convex optimization techniques, 5G communication networks, data encryption, among other emerging techniques. Lastly and most importantly, data privacy issue can be well addressed by the federated learning framework so that mobile users dare to share their location related information with safeguard.

Acknowledgement

We would like to thank Wenbiao Guo and Ang Xie from Beijing Jiaotong University and Haole Chen from Wuhan University for their kind help with preparing an early draft of this work.

This work was mainly supported by the Natural Science Foundation of China (NSFC) under Grant 61701426, and in part by the National Key Research and Development Program of China under Grant 2018YFB1800800, in part by the Guangdong Research Project under Grant 2017ZT07X152 and Grant 00201501, in part by Shenzhen Fundamental Research Fund under Grant KQTD201503311441545, and in part by the National Natural Science Foundation of China under Grant 61571334.

References

[1] C. E. Rasmussen and C. I. K. Williams, *Gaussian Processes for Machine Learning*, vol. 1, Cambridge, MA, USA: MIT Press, 2006.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[3] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, San Diego, CA, USA, May 2015.

[4] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, "Learning human identity from motion patterns," *IEEE Access*, vol. 4, pp. 1810–1820, Apr. 2016.

[5] N. H Tran, W. Bao, A. Zomaya, Nguyen M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 1387–1395.

[6] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Abu Dhabi, United arab emirates, Dec. 2018, pp. 1–7.

[7] J. Lee, J. Sun, F. Wang, S. Wang, C-H Jun, and X. Jiang, "Privacy-preserving patient similarity learning in a federated environment: development and analysis," *JMIR Med. Inform.*, vol. 6, no. 2, pp. e20, 2018.

[8] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: Possibilities and fundamental limitations based on available wireless network measurements," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 41–53, June 2005.

[9] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: Challenges faced in developing techniques for accurate wireless location information," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 24–40, July 2005.

[10] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley & Sons, Inc., New York, NY, 2001.

[11] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless traffic prediction with scalable Gaussian process: Framework, algorithms, and verification," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1291–1306, June 2019.

[12] L. Liu, Z. Qiu, G. Li, Q. Wang, W. Ouyang, and L. Lin, "Contextualized spatial–temporal network for taxi origin-destination demand prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3875–3887, May 2019.

[13] T. Y Kim and S. B Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy*, vol. 182, pp. 72–81, Sep. 2019.

[14] Z. Qi, T. Wang, G. Song, W. Hu, X. Li, and Z. Zhang, "Deep air learning: Interpolation, prediction, and feature analysis of fine-grained air quality," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2285–2297, Apr. 2018.

[15] A. Greenberg, "How apple and google are enabling covid-19 contact-tracing," https://www.wired.com/story/apple-google-bluetooth-contact-tracing-covid-19/.

[16] N. Guha, A. Talwlkar, and V. Smith, "One-shot federated learning," *arXiv preprint arXiv:1902.11175*, 2019.

[17] F. Sattler, S. Wiedemann, K. Mller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, Nov. 2019.

[18] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, United states, June 2019, pp. 634–643.

[19] E. Bagdasaryan, A.s Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Online, Aug. 2020, pp. 2938–2948.

[20] H. Kim, J. Park, M. Bennis, and S-L Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, pp. 1–4, June 2019.

[21] Q. Yang, Y. Liu, T. Chen, and Yong Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 492–503, Feb. 2019.

[22] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[23] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[24] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Trans. Sen. Netw.*, vol. 2, no. 2, pp. 188–220, May 2006.

[25] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.

[26] Y. Shen, H. Wymeersch, and M. Z. Win, "Fundamental limits of wideband localization—Part II: Cooperative networks," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 4981–5000, Sept. 2010.

[27] M. Z. Win, A. Conti, S. Mazuelas, Y. Shen, W. M. Gifford, D. Dardari, and M. Chiani, "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011.

[28] M. Z. Win, A. Conti, S. Mazuelas, Y. Shen, W. M. Gifford, D. Dardari, and M. Chiani, "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011.

[29] F. Yin, C. Fritsche, D. Jin, F. Gustafsson, and A. M. Zoubir, "Cooperative localization in WSNs using Gaussian mixture modeling: Distributed ECM algorithms," *IEEE Trans. Signal Process.*, vol. 63, no. 6, pp. 1448–1463, Mar. 2015.

[30] D. Jin, F. Yin, C. Fritsche, F. Gustafsson, and A. M. Zoubir, "Bayesian cooperative localization using received signal strength with unknown path loss exponent: Message passing approaches," *IEEE Trans. Signal Process.*, vol. 68, pp. 1120–1135, Jan. 2020.

[31] N. Patwari, A. O. Hero III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2137–2148, Aug. 2003.

[32] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Proc. Conf. Uncertain. Artif. Intell. (UAI)*, San Francisco, CA, USA, Aug. 2002, pp. 493–500.

[33] D. Gu, "Distributed particle filter for target tracking," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA)*, Rome, Italy, Apr. 2007, pp. 3856–3861.

[34] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *Proc. IEEE Conf. Decis. Control. (CDC)*, New Orleans, LA, USA, Dec. 2007, pp. 5492–5498.

[35] M. Kamgarpour and C. Tomlin, "Convergence properties of a decentralized kalman filter," in *Proc. IEEE Conf. Decis. Control. (CDC)*, Cancun, Mexico, Dec. 2008, pp. 3205–3210.

[36] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 444–457, Feb. 2015.

[37] C. Zhang, K. P. Subbu, J. Luo, and J. Wu, "GROPING: Geomagnetism and crowdsensing powered indoor navigation," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 387–400, Feb. 2015.

[38] E. Arias-de Reyna, D. Dardari, P. Closas, and P. M. Djuric, "Estimation of spatial fields of NLOS/LOS conditions for improved localization in indoor environments," in *IEEE Stat. Signal Process. Workshop (SSP)*, Freiburg, Germany, June 2018, pp. 658–662.

[39] P. J. Diggle, *Statistical analysis of spatial and spatio-temporal point patterns*, Chapman and Hall/CRC, third edition, 2013.

[40] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: A survey of problems and methods," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–41, Aug. 2018.

[41] S. Wang, J. Cao, and P. S. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, Sept. 2019.

[42] S. Sarkka, A. Solin, and J. Hartikainen, "Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering," *IEEE Signal Process. Mag.*, vol. 30, no. 4, pp. 51–61, June 2013.

[43] Y. Kuang, T. Chen, F. Yin, and R. Zhong, "Recursive implementation of gaussian process regression for spatial-temporal data modeling," in *Proc. Int. Conf. Wirel. Commun. Signal Process. (WCSP)*, Xi'an, China, Oct. 2019, pp. 1–7.

[44] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[45] Y. E. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983.

[46] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, July 2011.

[47] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning," *Technical Report*, 2017.

[48] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, United states, May 2015.

[49] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

[50] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Sardinia, Italy, May 2010, pp. 249–256.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1026–1034.

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Lile, France, July 2015, pp. 448–456.

[53] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Barcelona, Spain, Dec. 2016, pp. 901–909.

[54] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," in *Proc. NeurIPS Deep Learning Symposium*, Barcelona, Spain, Dec. 2016.

[55] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[56] R. Sun, "Optimization for deep learning: theory and algorithms," *arXiv preprint arXiv:1912.08957*, 2019.

[57] S. Theodoridis, *Machine Learning: a Bayesian and Optimization Perspective*, Academic Press, 2nd edition, 2020.

[58] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA,USA: MIT Press, 2016.

[59] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, Paris, France, June 2010, pp. 253–256.

[60] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, California, USA, Dec. 2017, pp. 3856–3866.

[61] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017.

[62] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NeurIPS Deep Learning and Representation Learning Workshop*, Montreal, QC, Canada, Dec. 2015.

[63] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019.

[64] A. G. Wilson and R. P. Adams, "Gaussian process kernels for pattern discovery and extrapolation," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Atlanta, USA, July 2013, pp. 1067–1075.

[65] F. Yin, X. He, L. Pan, T. Chen, Z.-Q. Luo, and S Theodoridis, "Sparse structure enabled grid spectral mixture kernel for temporal Gaussian process regression," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Cambridge, UK, July 2018, pp. 47–54.

[66] F. Yin, L. Pan, T. Chen, S. Theodoridis, Z.-Q. Luo, and A. M. Zoubir, "Linear multiple low-rank kernel based stationary gaussian processes regression for time series," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5260 – 5275, Sept. 2020.

[67] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Cadiz, Spain, May 2016, pp. 370–378.

[68] A.G Wilson, Z Hu, R Salakhutdinov, and E. P Xing, "Stochastic variational deep kernel learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Barcelona, Spain, Dec. 2016, pp. 2586–2594.

[69] M. Al-Shedivat, A. G. Wilson, Y. Saatchi, Z. Hu, and E. P Xing, "Learning scalable deep kernels with recurrent structure," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2850–2886, Aug. 2017.

[70] H. Xue, Z-F Wu, and W-X Sun, "Deep spectral kernel learning," in *Proc. IJCAI Int. Joint Conf. Artif. Intell. (IJCAI)*, Macao, China, Aug. 2019, pp. 4019–4025.

[71] A. Damianou and N. Lawrence, "Deep Gaussian processes," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Scottsdale, AZ, USA, Apr. 2013, pp. 207–215.

[72] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 2009, pp. 342–350.

[73] A. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani, "Gaussian process behaviour in wide deep neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr. 2018.

[74] J. Lee, J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, "Deep neural networks as Gaussian processes," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr. 2018.

[75] C. K. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, December 2001, pp. 682–688.

[76] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W Hogg, and M. ONeil, "Fast direct methods for Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 252–265, Feb. 2015.

[77] M. P. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Lille, France, July 2015, pp. 1481–1490.

[78] M. K. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Clearwater Beach, Florida, USA, Apr. 2009, pp. 567–574.

[79] H. Liu, Y-S Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–19, Jan. 2020.

[80] R. M. Neal, *Bayesian Learning for Neural Networks*, Ph.D. thesis, University of Toronto, Canada, 1995.

[81] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, Canada, Dec. 2018, pp. 8571–8580.

[82] S. Arora, SS Du, W. Hu, Z. Li, RR Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 2019, pp. 8139–8148.

[83] A. Girard, *Approximate Methods for Propagation of Uncertainty with Gaussian Process Model*, Ph.D. thesis, Univerity of Glasgow, Glasgow, UK, 2004.

[84] C. Bishop, *Machine Learning and Pattern Recognition*, New York, USA: Springer, 2006.

[85] Simo Särkkä, *Bayesian Filtering and Smoothing*, Cambridge University Press, 2013.

[86] M. Loog, T. Viering, A. Mey, J. H. Krijthe, and D. M. J. Tax, "A brief prehistory of double descent," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 117, no. 20, pp. 10625–10626, 2020.

[87] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical biasvariance trade-off," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 116, no. 32, pp. 15849–15854, Aug. 2019.

[88] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Online, Apr. 2020.

[89] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," *arXiv preprint arXiv:2002.08791*, 2020.

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

[91] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[92] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optimiz.*, vol. 26, no. 1, pp. 337–364, Jan. 2016.

[93] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[94] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[95] C.-H. Fang, S. B. Kylasa, F. Roosta-Khorasani, M. W. Mahoney, and A. Grama, "Newton-ADMM: A distributed GPU-accelerated optimizer for multiclass classification problems," *arXiv preprint arXiv:1807.07132v3*, 2020.

[96] Y. Tang, J. Zhang, and N. Li, "Distributed zero-order algorithms for nonconvex multi-agent optimization," *IEEE Trans. Control Netw. Syst.*, pp. 1–12, Sept. 2020.

[97] P. Richtárik and M. Takáč, "Parallel coordinate descent methods for big data optimization," *Math. Program.*, vol. 156, no. 1-2, pp. 433–484, Mar. 2016.

[98] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Inference attacks against collaborative learning," *arXiv preprint arXiv:1805.04049*, 2018.

[99] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Barcelona, Spain, Dec. 2016.

[100] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM Conf. Computer Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 1175–1191.

[101] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 911–926, July 2020.

[102] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 2031–2063, Apr. 2020.

[103] RC. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[104] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[105] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proc. ACM Conf. Computer Commun. Secur.*, New York, NY, USA, Nov. 2019, pp. 1–11.

[106] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Advances in Cryptology—EUROCRYPT 1999*, Berlin, Heidelberg, May 1999, pp. 223–238.

[107] E. Mohyeldin, "Minimum technical performance requirements for imt-2020 radio interface(s)," https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Documents/S01-1_Requirements%20for%20IMT-2020_Rev.pdf, 2020.

[108] "3gpp release 10," https://www.3gpp.org/specifications/releases/70-release-10, 2013.

[109] "IEEE draft standard for information technology – telecommunications and information exchange between systems local and metropolitan area networks – specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment enhancements for high efficiency wlan," *IEEE P802.11ax/D6.0, November 2019*, pp. 1–780, Dec 2019.

[110] "IEEE standard for information technology– telecommunications and information exchange between systems local and metropolitan area networks– specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz.," *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)*, pp. 1–425, Dec 2013.

[111] Y. Xu, F. Yin, W. Xu, C-H Lee, J. Lin, and S. Cui, "Scalable learning paradigms for data-driven wireless communication," *arXiv preprint arXiv:2003.00474*, March 2020.

[112] F. Gustafsson and F. Gunnarsson, "Measurements used in wireless sensor networks localization," in *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking*, pp. 33–53. IGI Global, Hershey, PA, USA, 2009.

[113] F. Yin, Y. Zhao, and F. Gunnarsson, "Proximity report triggering threshold optimization for network-based indoor positioning," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Washington, DC, USA, July 2015, pp. 1061–1069.

[114] R. P. Ghozali and G. P. Kusuma, "Indoor positioning system using regression-based fingerprint method," *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 10, no. 8, pp. 231–239, 2019.

[115] A. Sahar and D. Han, "An LSTM-Based indoor positioning method using Wi-Fi signals," in *Proc. Int. Conf. Vis., Image Signal Process. (ICVISP)*, Las Vegas, NV, USA, Aug. 2018, pp. 1–5.

[116] J. Liu, N. Liu, Z. Pan, and X. You, "Autloc: Deep autoencoder for indoor localization with RSS fingerprinting," in *Proc. Int. Conf. Wirel. Commun. Signal Process. (WCSP)*, Hangzhou, China, Oct. 2018, pp. 1–6.

[117] C.-H. Hsieh, J.-Y. Chen, and B.-H. Nien, "Deep learning-based indoor localization using received signal strength and channel state information," *IEEE Access*, vol. 7, pp. 33256–33267, Mar. 2019.

[118] X. Wang, X. Wang, and S. Mao, "Deep convolutional neural networks for indoor localization with CSI images," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 316–327, 2020.

[119] A.P. James, "Towards strong AI with analog neural chips," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sevilla, Spain, Oct. 2020, pp. 1–5.

[120] J. Qi, H. Li, F. Yin, B. Ai, and S. Cui, "Navigation with low-sampling-rate GPS and smartphone sensors: A data-driven learning-based approach," in *Proc. IET Int. Conf. Wireless, Mobile and Multimedia Netw. (ICWMMN)*, Beijing, China, Nov. 2019, pp. 1–6.

[121] T. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, Jan. 2011.

[122] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *J. Roy. statistical Soc. B-Stat. Method.*, vol. 72, no. 3, pp. 269–342, June 2010.

[123] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational Gaussian process state-space models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, Canada, Dec. 2014, pp. 3680–3688.

[124] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proc. IJCAI Int. Joint Conf. Artif. Intell. (IJCAI)*, Hyderabad, India, Jan. 2007, pp. 2480–2485.

[125] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, Feb. 2008.

[126] J. Ko and D. Fox, "Learning gp-bayesfilters via gaussian process latent variable models," *Auton. Robots*, vol. 30, no. 1, pp. 3–23, Jan. 2011.

[127] R. Frigola, F. Lindsten, T. B Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Lake Tahoe, Nevada, USA, Dec. 2013, pp. 3156–3164.

[128] S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman, "Identification of Gaussian process state space models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, California, USA, Dec. 2017, pp. 5309–5319.

[129] A. D. Ialongo, M. van der Wilk, and C. E. Rasmussen, "Closed-form inference and prediction in Gaussian process state-space models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS) Time Series Workshop*, Long Beach, California, USA, Dec. 2017.

[130] A. D. Ialongo, M. van der Wilk, J. Hensman, and C. E. Rasmussen, "Non-factorised variational inference in dynamical systems," in *Proceedings of Symposium on Advances in Approximate Bayesian Inference*, Montreal, Canada, Dec. 2018.

[131] Y. Zhao, F. Yin, F. Gunnarsson, F. Hultkratz, and J. Fagerlind, "Gaussian processes for flow modeling and prediction of positioned trajectories evaluated with sports data," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Heidelberg, Germany, July 2016, pp. 1461–1468.

[132] F. Yin and F. Gunnarsson, "Distributed recursive Gaussian processes for RSS map applied to target tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 492–503, Apr. 2017.

[133] F. Gustafsson, *Statistical Sensor Fusion*, Studentlitteratur, Lund, Sweden, 2012.

[134] A. Xie, F. Yin, B. Ai, S. Zhang, and S. Cui, "Learning while tracking: A practical system based on variational Gaussian process state-space model and smartphone sensory data," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Rustenburg, South Africa, July 2020, pp. 1–7.

[135] Y. Zhao, C. Fritsche, F. Yin, and F. Gunnarsson, "Cramer-rao bounds for filtering based on Gaussian process state-space models," *IEEE Trans. Signal Process.*, vol. 67, no. 23, pp. 5936–5951, Dec. 2019.

[136] Y. Gal, M. van der Wilk, and C. E. Rasmussen, "Distributed variational inference in sparse Gaussian process regression and latent variable models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, Canada, Dec. 2014, pp. 3257–3265.

[137] R. Senanayake, S. O'Callaghan, and F. Ramos, "Predicting spatio-temporal propagation of seasonal influenza using variational Gaussian process regression," in *Proc. AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, Feb. 2016, pp. 3901–3907.

[138] E. V Bonilla, K. M. Chai, and C. Williams, "Multi-task gaussian process prediction," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pp. 153–160. Vancouver, BC, Canada, Dec. 2008.

[139] M. Van Der Wilk, C. E. Rasmussen, and J. Hensman, "Convolutional gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pp. 2849–2858. Long Beach, CA, USA, Dec. 2017.

[140] Y. C. Ng, N. Colombo, and R. Silva, "Bayesian semi-supervised learning with graph Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, Canada, Dec. 2018, pp. 1690–1701.

[141] I. Walker and B. Glocker, "Graph convolutional Gaussian processes," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Long Beach, California, USA, June 2019, pp. 6495–6504.

[142] Y. Zhao, C. Liu, L. S. Mihaylova, and F. Gunnarsson, "Gaussian processes for RSS fingerprints construction in indoor localization," in *Proc. Int. Conf. Inf. Fusion (FUSION)*, Cambridge, UK, Sept. 2018, pp. 1377–1384, IEEE.

[143] Z. Li, L. Wang, L. Jiang, and C. Xu, "FC-SLAM: Federated learning enhanced distributed visual-LiDAR SLAM in cloud robotic system," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dali, China, Dec. 2019, pp. 1995–2000.

[144] J. Waterston, J. Rhea, S. Peterson, L. Bolick, J. Ayers, and J. Ellen, "Ocean of things : Affordable maritime sensors with scalable analysis," in *OCEANS 2019 - Marseille*, Marseille, France, Oct. 2019, pp. 1–6.

[145] H. Chen, D. Li, Y. Wang, and F. Yin, "UAV Hovering Strategy Based on a Wirelessly Powered Communication Network," *IEEE Access*, vol. 7, pp. 3194–3205, Dec. 2018.

[146] Y. Hu, X. Yuan, J. Xu, and A. Schmeink, "Optimal 1d trajectory design for uav-enabled multiuser wireless power transfer," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5674–5688, Apr. 2019.

[147] F. Yin, Y. Zhao, F. Gunnarsson, and F. Gustafsson, "Received-signal-strength threshold optimization using Gaussian processes," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2164–2177, Apr. 2017.

[148] K. Bonawitz, H. Eichner, W. Grieskamp, and D. Huba et.al., "Towards federated learning at scale: System design," in *Proc. SysML Conf.*, Palo Alto, CA, USA, Mar. 2019.

[149] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, July 2018, pp. 560–569.

[150] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*, Princeton University Press, 2009.

[151] B. L Gorissen, İ. Yanıkoğlu, and D. den Hertog, "A practical guide to robust optimization," *Omega*, vol. 53, pp. 124–137, June 2015.

[152] A. M. Zoubir, V. Koivunen, E. Ollila, and M. Muma, *Robust Statistics for Signal Processing*, Cambridge University Press, 2018.

[153] A. Zappone, M. Di Renzo, M. Debbah, T. T. Lam, and X. Qian, "Model-aided wireless artificial intelligence: Embedding expert knowledge in deep neural networks for wireless system optimization," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 60–69, Sept. 2019.

[154] P. Hu, Z. Yan, R. Huang, and F. Yin, "How effectively can indoor wireless positioning relieve visual tracking pains: A Cramer-Rao bound viewpoint," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taipei, Taiwan, Sept. 2019, pp. 3083–3087.

[155] H. Ahmadi, A. Polo, T. Moriyama, M. Salucci, and F. Viani, "Semantic wireless localization of WiFi terminals in smart buildings," *Radio Sci.*, vol. 51, no. 6, pp. 876–892, June 2016.

[156] S. Guo, H. Xiong, X. Zheng, and Y. Zhou, "Activity recognition and semantic description for indoor mobile localization," *Sensors*, vol. 17, no. 3, pp. 649, Mar. 2017.

Feng Yin is currently an assistant professor in the School of Science and Engineering at the Chinese University of Hong Kong, Shenzhen, China. He received the B.Sc. degree from Shanghai Jiao Tong University, China, in 2008, and the M.Sc. and Ph.D. degrees from Technische Universität Darmstadt, Germany, in 2011 and 2014, respectively. From 2014 to 2016, he was with Ericsson Research, Linkoping, Sweden, working on the European Union FP7 Marie Curie Training Programme on Tracking in Complex Sensor Systems (TRAX). Since 2016, he has been with The Chinese University of Hong Kong, Shenzhen and also affiliated with the Shenzhen Research Institute of Big Data (SRIBD). His research interests include statistical signal processing, Bayesian deep learning, and sensory data fusion. He was a recipient of the Chinese Government Award for Outstanding Self-Financed Students Abroad in 2013. He received the Marie Curie Scholarship from the European Union in 2014. He is currently serving as the handling editor of the Elsevier Signal Processing and Elsevier Digital Signal Processing.

Zhidi Lin received the M.Sc. degree in communication and information systems from Xiamen University, Xiamen, China in 2019. He is currently pursuing the Ph.D. degree with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His research interests lie in the areas of statistical signal processing, Bayesian deep learning, and related fields.

Yue Xu received his B.S. and Ph.D. degree from Beijing University of Post and Telecommunication (BUPT) in 2020. He has been a Visiting Researcher with University of California, Davis, USA, The Chinese University of Hong Kong, Shenzhen, China, and Shenzhen Research Institute of Big Data. He is currently a research scientist at Alibaba Group. His research interests include data-driven wireless network management, machine learning, large-scale data analytics and system control.

Qinglei Kong received her PhD degree from the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2018; the M.Eng. degree in electronic and information engineering from Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China, in 2015; and the B.Eng. degree in communication engineering from Harbin Institute of Technology, Harbin, China, in 2012. She used to work in Cyber Security Cluster, Institute for Infocomm Research, Singapore and Tencent Security, Shenzhen, as a research scientist. Now she is working in The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), as a postdoc researcher. Her research interests include applied cryptography, blockchain, VANET, and game theory.

Deshi Li received his PhD degree in Computer Application Technology from Wuhan University. He was a visiting scholar of the Network Lab of the University of California at Davis.

He is a professor of Electronic Information School, Wuhan University. His research interests include wireless communication, Internet of Things, intelligence system and SOC design. He has published more than 100 research papers. He serves as a reviewer for many international academic journals and an expert evaluator for the Ministry of Science and Technology of China, Ministry of Education of China and NSF China.

Currently Dr. Li serves as a member of the Internet of Things Expert Committee and member of the Education Committee of Chinese Institute of Electronics, and the Associate Chief scientist in Space Communication area of Collaborative Innovation Center of Geospatial Technology, also is an Executive Trustee member of China Cloud System Pioneer Strategic Alliance. His recent research projects include National Science and Technology Major Project of China (973 Program), National High Technology Program of China (863 Program), and National Natural Science Foundation of China (NSFC).

Sergios Theodoridis is Professor Emeritus of Signal Processing and Machine Learning in the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens, Greece and with the Shenzhen Research Institute of Big Data (SRIBD), the Chinese University of Hong Kong, Shenzhen, China. His research interests lie in the areas of Online Algorithms, Distributed and Sparsity-Aware Learning, Machine Learning, Signal Processing and Learning for Bio-Medical Applications and Audio Processing and Retrieval.

He is the author of the book "Machine Learning: A Bayesian and Optimization Perspective" Academic Press, 2nd Ed., 2020, the co-author of the best-selling book "Pattern Recognition", Academic Press, 4th ed. 2009, the co-author of the book "Introduction to Pattern Recognition: A MATLAB Approach", Academic Press, 2010, the co-editor of the book "Efficient Algorithms for Signal Processing and System Identification", Prentice Hall 1993, and the co-author of three books in Greek, two of them for the Greek Open University.

He is the co-author of seven papers that have received Best Paper Awards including the 2014 IEEE Signal Processing Magazine Best Paper award and the 2009 IEEE Computational Intelligence Society Transactions on Neural Networks Outstanding Paper Award.

He is the recipient of the 2017 EURASIP Athanasios Papoulis Award, the 2014 IEEE Signal Processing Society Education Award and the 2014 EURASIP Meritorious Service Award. He has served as a Distinguished Lecturer for the IEEE Signal Processing as well as the Circuits and Systems Societies. He was Otto Monstead Guest Professor, Technical University of Denmark, 2012, and holder of the Excellence Chair, Dept. of Signal Processing and Communications, University Carlos III, Madrid, Spain, 2011.

He currently serves as Vice President IEEE Signal Processing Society. He has served as President of the European Association for Signal Processing (EURASIP), as a member of the Board of Governors for the IEEE Circuits and Systems (CAS) Society, as a member of the Board of Governors (Member-at-Large) of the IEEE SP Society and as a Chair of the Signal Processing Theory and Methods (SPTM) technical committee of IEEE SPS

He has served as Editor-in-Chief for the IEEE Transactions on Signal Processing. He is Editor-in-Chief for the Signal Processing Book Series, Academic Press and co-Editor in Chief for the E-Reference Signal Processing, Elsevier.

He is Fellow of IET, a Corresponding Fellow of the Royal Society of Edinburgh (RSE), a Fellow of EURASIP and a Life Fellow of IEEE.

Shuguang (Robert) Cui received his Ph.D in Electrical Engineering from Stanford University, California, USA, in 2005. Afterwards, he has been working as assistant, associate, full, Chair Professor in Electrical and Computer Engineering at the Univ. of Arizona, Texas A&M University, UC Davis, and CUHK at Shenzhen respectively. He has also been the Vice Director at Shenzhen Research Institute of Big Data. His current research interests focus on data driven large-scale system control and resource management, large dataset analysis, IoT system design, energy harvesting based communication system design, and cognitive network optimization. He was selected as the Thomson Reuters Highly Cited Researcher and listed in the Worlds Most Influential Scientific Minds by ScienceWatch in 2014. He was the recipient of the IEEE Signal Processing Society 2012 Best Paper Award. He has served as the general co-chair and TPC co-chairs for many IEEE conferences. He has also been serving as the area editor for IEEE Signal Processing Magazine, and associate editors for IEEE Transactions on Big Data, IEEE Transactions on Signal Processing, IEEE JSAC Series on Green Communications and Networking, and IEEE Transactions on Wireless Communications. He has been the elected member for IEEE Signal Processing Society SPCOM Technical Committee (2009 2014) and the elected Chair for IEEE ComSoc Wireless Technical Committee (2017 2018). He is a member of the Steering Committee for IEEE Transactions on Big Data and the Chair of the Steering Committee for IEEE Transactions on Cognitive Communications and Networking. He was also a member of the IEEE ComSoc Emerging Technology Committee. He was elected as an IEEE Fellow in 2013 and an IEEE ComSoc Distinguished Lecturer in 2014.